НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«До захисту допущено»

Завідувач кафедри

_____ О.І. Ролік

«___»_____20__ р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050201 «Системна інженерія»

на тему: «Веб-застосунок для дистанційного замовлення їжі»

Виконав:
Студент IV курсу, групи ІА-52
Чекайда Влас Віталійович _____

Керівник:
Доцент, к.т.н., доцент кафедри АУТС Амонс О. А.
_____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2019 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050201 «Системна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.І. Ролік

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проект студенту

Чекайди Власа Віталійовича

1. Тема проекту «Веб-застосунок для дистанційного замовлення їжі», керівник проекту Амонс Олександр Анатолійович, доцент, к.т.н., затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту

_____

_____

4. Зміст пояснювальної записки

_____

_____

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

_____

_____

_____

6. Дата видачі завдання_____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення предметної області	До 17.04.2019	
2	Аналіз існуючих рішень	До 8.05.2019	
3	Побудова діаграм	До 14.05.2019	
4	Вибір архітектури і платформи	До 17.05.2019	
5	Написання коду для веб-проекту	До 31.05.2019	
6	Оформлення текстової документації	До 14.06.2019	

Студент В. В. Чекайда

Керівник проекту О. А. Амонс

АНОТАЦІЯ

Чекайда В. В. Веб-застосунок для дистанційного замовлення їжі. НТУУ «КПІ ім. Сікорського», Київ, 2019.

Проект містить 70 с. тексту, 48 рисунків, 1 таблиця, посилання на 13 літературних джерел.

Ключові слова: Мережа Інтернет, веб-сайт, база даних, фреймворк, програмне забезпечення.

Об'єктом розробки є веб-застосунок.

Мета роботи – можливість надання сервісу для замовлення їжі через браузер.

У дипломному проекті розроблено модель системи, за допомогою якої користувач має змогу замовити їжу. Було обрано сучасні технології для створення веб-проекту та спроектовано загальну модель.

Проект може бути корисним підґрунтям для підприємців, які мають ресторан, кафе або магазин і хочуть мати свій веб-сервіс для того, щоб люди дистанційно замовляли їжу, продукти тощо.

ANNOTATION

Chekaida V. V. Web-application for online food ordering. NTUU “Igor Sikorsky KPI”, Kyiv, 2019.

The project contains 70 page text, 48 figures, 1 table, links to 13 literary sources.

Keywords: Internet, web-site, database, framework, software.

The object of development is a web-application.

The purpose of the work is to provide service for ordering food with help of browser.

In the diploma project has developed model. It makes able user to order food. Modern technologies were chosen for creating web-project and general model was designed.

Project can be helpful for entrepreneurs, who have their own cafe, shop or restaurant and want to have web-service for remote ordering food, products, etc.

Номерсторінк	Формат	Позначення	Найменування	Кільк. листів	Номер екзем.	Приміт-ка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА52.030БАК.005 ПЗ	Пояснювальна записка	70		
6	А3	ІА52.030БАК.005 Д1	Діаграма прецедентів	1		
7						
8	А3	ІА52.030БАК.005 Д2	Класова діаграма	1		
9						
10	А3	ІА52.030БАК.005 Д3	Діаграма активності	1		
11						
12	А3	ІА52.030БАК.005 Д4	Карта сайту	1		
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Арк.	№ докум.	Підпис	Дата	ІА52.300БАК.005ТП		
Розробив		Чекайда В. В.			Веб-застосунок для дистанційного замовлення їжі Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Амонс О. А.				Т	
Рецен.						КПІ ім. Ігоря Сікорського, ФІОТ Група ІА-52	
Н. Контр.							
Затв.							

Пояснювальна записка

до дипломного проекту

на тему: «Веб-застосунок для дистанційного замовлення їжі»

Київ – 2019 рік

ЗМІСТ

ВСТУП 5

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ 7

1.1 Аналіз існуючих веб-сервісів для доставки їжі 7

1.1.1 eda.ua 7

1.1.2 "Экипаж сервис" 7

1.1.3 Glovo 9

1.1.4 Royal Service 10

1.1.5 "Суши 33" і Pizza 33 11

1.1.6 "Наша карта" 12

1.3 Висновок 12

2 ВИБІР АРХІТЕКТУРИ І ПЛАТФОРМИ 14

2.1 Основні поняття 14

2.2 Клієнт-сервер 14

2.3 Мова програмування 15

2.4 Середовище розробки 16

2.5 Висновки 17

3 СУЧАСНІ ТЕХНОЛОГІЇ 18

3.1 Основні поняття 18

3.2 Методи створення сайтів 18

3.2.1 Ручна за допомогою HTML та CSS 19

3.2.2 За допомогою програмних засобів 20

3.2.3 Content Management System 21
3.3 Фреймворк Spring 22
3.3.1 IoC-контейнер 23
3.3.2 Model-View-Controller 24
3.3.3 Data 26
3.3.4 Boot 28
3.4 Hibernate 28
3.5 Maven 30
3.6 База даних 31
3.7 JSP 32
3.8 HTTP 33
3.8.1 GET запит 33
3.8.2 POST запит 33
3.9 MessageDigest5 34
3.10 Висновки 34
4 ПРОЕКТУВАННЯ СИСТЕМИ 36
4.1 Unified Modeling Language 36
4.2 Функціональна модель 36
4.3 Концептуальна модель 40
4.4 Загальний опис роботи системи 40
4.4.1 Загальна діаграма діяльності 46
4.4.2 Діаграма діяльності адміністратора 48
4.5 Карта сайту 49
4.6 Висновки 52
5 ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ 53

5.1 Гість 53
5.2 Користувач 58
5.3 Кухар 59
5.4 Кур'єр 63
5.5 Адміністратор 64
5.6 Висновок 67
ВИСНОВКИ 68
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 69

ВСТУП

У сучасному світі дуже стрімко розвиваються інформаційні технології. Одним із найбільших досягнень людини - Інтернет. У зв'язку зі швидким зростанням користувачів в мережі Інтернет, збільшується обсяг інформації та ресурси, які цю інформацію надають.

Інтернет - всесвітня мережа, яка об'єднала різного роду мережі та звичайних користувачів у величезну систему для обміну інформацією. Обмін інформацією здійснюється на базі протоколів TCP/IP.

Інформація в Інтернеті знаходиться на веб-сайтах. Веб-сайт - зв'язані між собою веб-сторінки, які мають спільну адресу, URL-адрес. Веб-сторінка - інформаційний документ, який показує браузер, веб-сторінка частіше за все - це HTML.

Коли тільки був створений Інтернет, ніхто і гадки не мав, що люди не тільки будуть обмінюватись інформацією, а і будувати, наприклад, свій власний бізнес. Підприємців, які виходять на інтернет-ринок стало дуже багато і с кожним роком їх все більше й більше. Чому ж люди обирають Інтернет. Відповідь на це питання дати дуже легко. По-перше, статистика про зростання користувачів мережею говорить нам, що с кожним роком все більше людей приєднуються до Інтернету, це означає, що в мережі з'являються нові ресурси, сервіси та багато іншого, що можуть нам допомогти у повсякденному житті і не тільки, по-друге, люди стали залежні від інформації. Ця залежність грає неймовірну роль у майбутньому розвитку інформаційних технологій.

На сьогоднішній день майже всі крупні фірми, організації, підприємства мають свій власний сайт. Сайти проектують і пишуть, як IT-гіганти, так і звичайні індивідуальні веб-розробники. Наприклад, щоб стати програмістом, не обов'язково ходити на курси, вчитися в університеті або мати друзів програмістів. Достатньо буде просто знайти в Інтернеті інформацію і навчитись її використовувати.

Кожна людина в середньому потребує 2-3 рази на день поїсти. Іноді складається так, що людина не в змозі поїсти вдома або на роботі, для цього існують так звані сервіси для доставки. У сучасному світі багато людей користуються цим сервісом саме через мережу Інтернет або через мобільні додатки. Саме тому, що їжа для людини являється і буде являтися невід'ємною частиною життєдіяльності, розробка технологій для замовлення їжі - це дуже актуальна тема сьогодення.

Метою цієї роботи є проектування і розробка некомерційного веб-сайту, за допомогою використання сучасних технологій. Задача сайту надати користувачу змогу замовити їжу у декілька кліків.

Сайт створений, як для клієнтів, так і для працюючих в компанії, підприємстві тощо. Тобто сайтом користуються кур'єри, кухарі, а також адміністратор.

Задачі:

1) Проаналізувати існуючі рішення;
2) Дослідити і засвоїти головні принципи побудови подібних веб-проектів;
3) Обрати оптимальну платформу для створення проекту;
4) Спроектувати і розібрати структуру веб-сайту;
5) Описати роботу системи та її процеси.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз існуючих веб-сервісів для доставки їжі

Список сервісів "деліверінгу":

- eda.ua
- "Экипаж сервис"
- Glovo
- Royal Service
- "Суши 33" и Pizza 33
- "Наша карта"

1.1.1 eda.ua

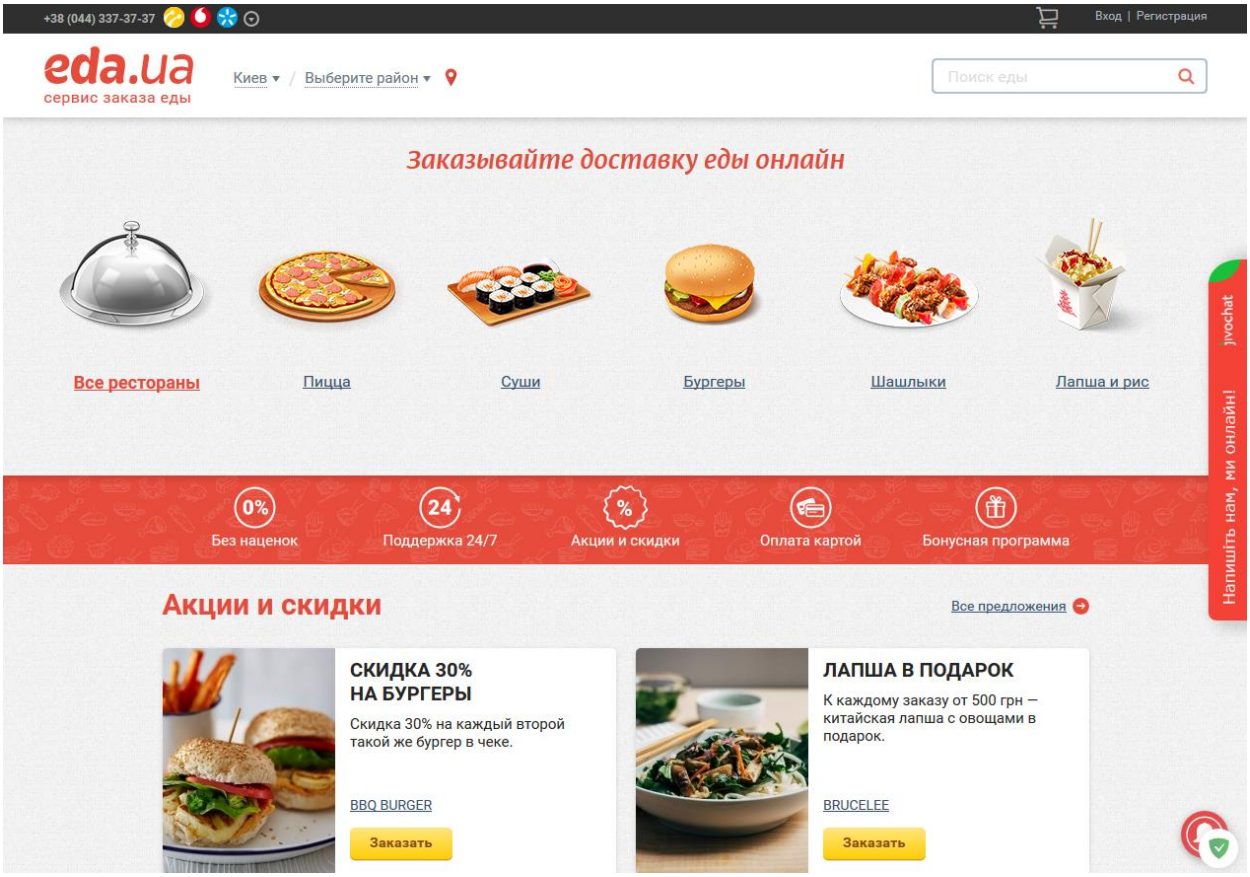


Рисунок 1.1 - Сайт eda.ua

Сайт: https://eda.ua/

Переваги: Великий асортимент, доставка по всій країні. Великий функціонал фільтрів. Підтримка різних платформ.

Недоліки: Сайт повільний, підтримує тільки російську мову. Час доставки може коливатись, через великий запит користувачів.

1.1.2 "Экипаж сервис"

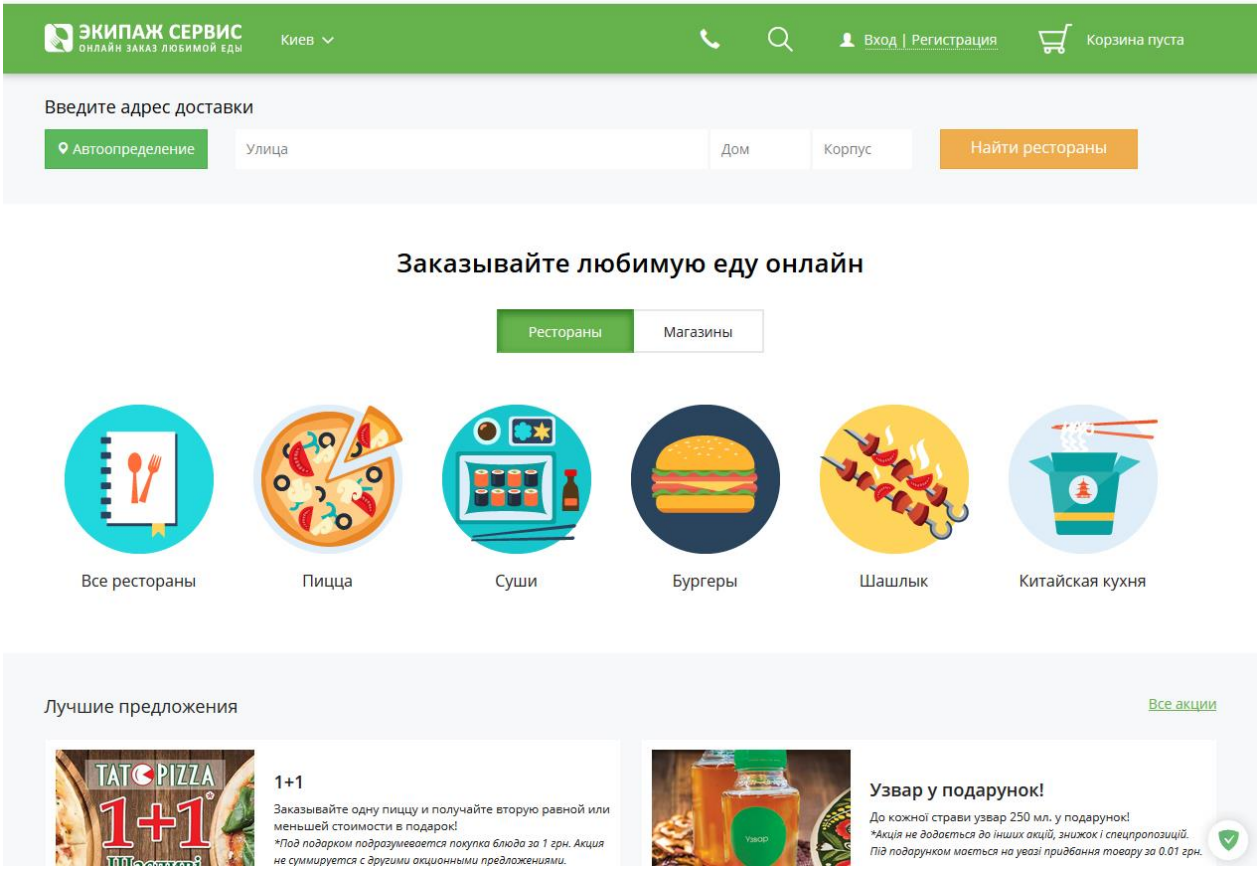


Рисунок 1.2 - Сайт "Экипаж Сервис"

Сайт: https://ekipazh-service.com.ua/

Переваги: Приємний у користуванні інтерфейс, підтримує понад 800 ресторанів і магазинів, доставка по всій країні. Великий функціонал фільтрів. Мобільний додаток та різні платформи. Знаходить ресторани і магазини поблизу.

Недоліки: Підтримує тільки російську мову. Швидкість середня.

1.1.3 Glovo

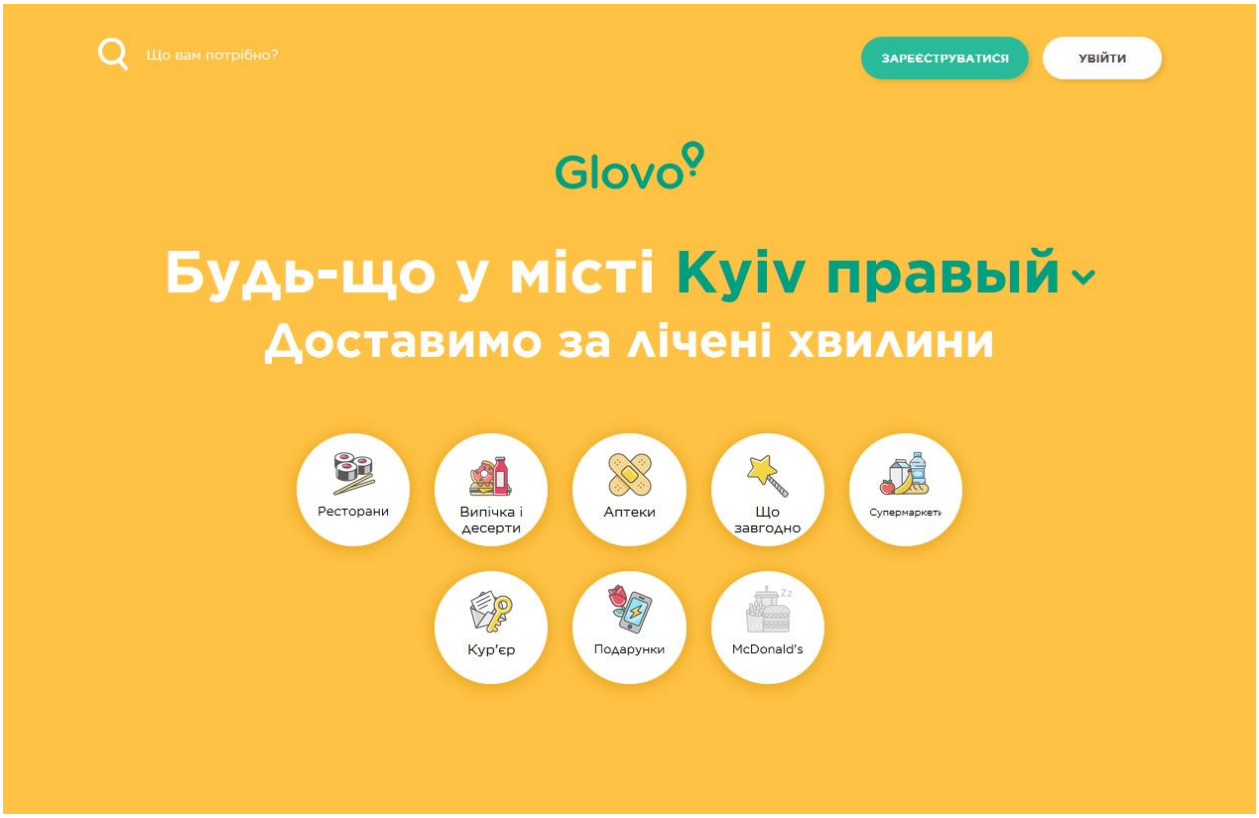


Рисунок 1.3 - Сайт Glovo

Сайт: https://glovoapp.com/uk/kie

Переваги: Дуже приємний у користуванні інтерфейс, швидкий сайт і сервіс, підтримка багатьох платформ. Підтримка понад 10 мов. Сайтом користуються як клієнти, так і працівники. Пошук найближчих ресторанів та маркетів.

Недоліки: Існує декілька незначних помилок у контенті, оскільки сервісу декілька місяців(2019 р.).

1.1.4 Royal Service

Рисунок 1.4 - Сайт Royal Service

Сайт: http://rsd.com.ua/

Переваги: Великий асортимент. Підтримка різних платформ. Швидкий сайт.

Недоліки: Підтримує тільки російську мову. Доставка по одному місту. Не дуже приємний інтерфейс. Реклама.

1.1.5 "Суши 33" і Pizza 33

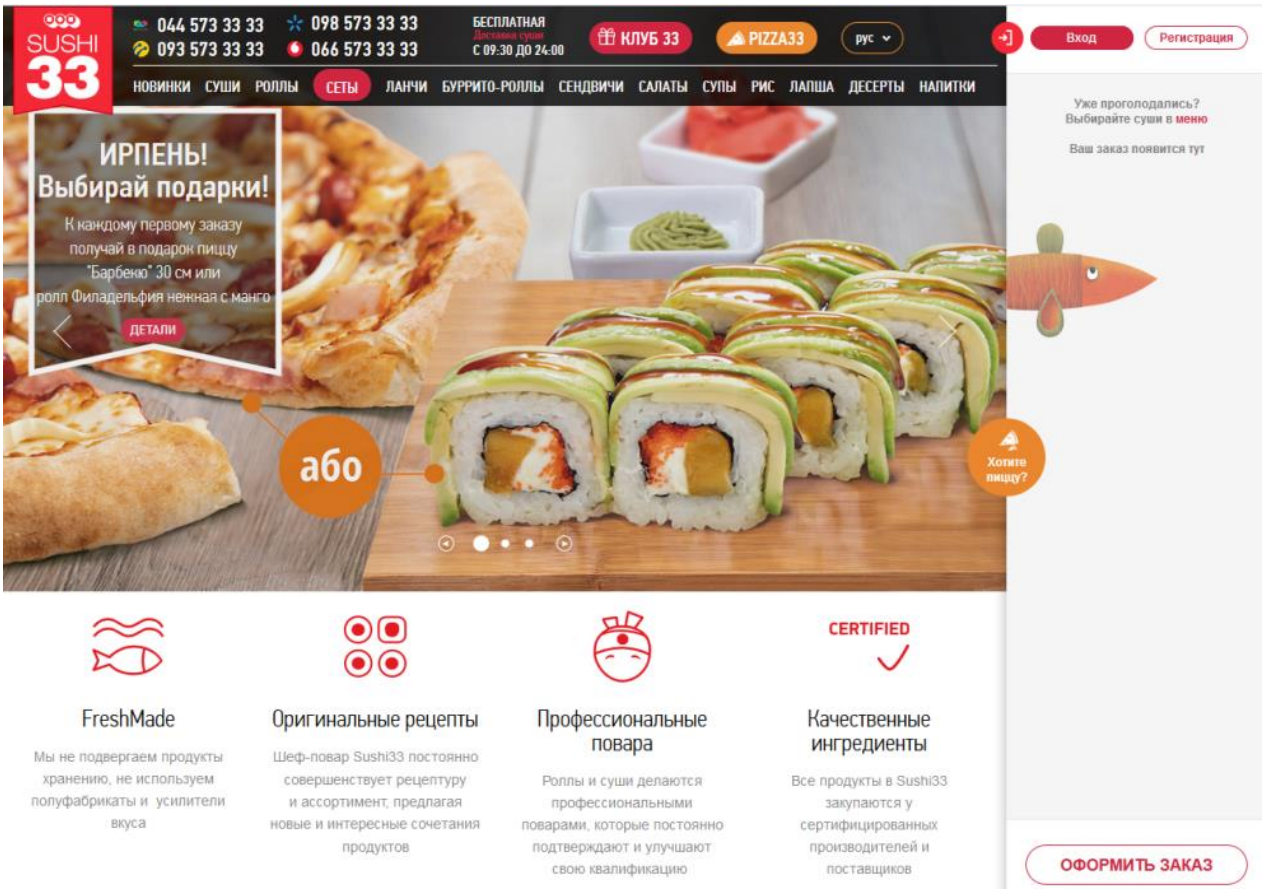


Рисунок 1.5 - Сайт "Суши 33" і Pizza 33

Сайт: https://sushi33.ua/

Переваги: Великий асортимент. Швидкий сайт. Підтримка трьох мов.

Недоліки: Специфічний інтерфейс. Не цілодобовий сервіс. Доставка по одному місту.

1.1.6 "Наша карта"

Рисунок 1.6 - Сайт "Наша карта"

Сайт: https://dostavka.nasha-karta.ua/listrest/

Переваги: Швидкий сайт. Підтримка двох мов.

Недоліки: Примітивний інтерфейс. Доставка по одному місту. Відсутній мобільний додаток.

1.3 Висновок

Проаналізувавши найбільш популярні веб-сайти, можна зробити висновок, що кожен із них має свої переваги і недоліки, але одним з найкращих сервісів на сьогоднішній день - це Glovo, спочатку було створено мобільний додаток, а вже потім веб-застосунок. У цьому веб-сайті є все, що

потрібно користувачі, щоб зручно використовувати цей сервіс: інтернаціоналізація, зручний інтерфейс, цілодобова підтримка, швидкість обслуговування, пошук необхідних ресторанів за місцезнаходженням інтернет-користувача, сервіс доставки всього, чого забажає клієнт, за умови об'єму і ваги товару. Glovo - компанія, заснована у 2015 році. В Україні з'явилась тільки у цьому році(2019 р.). Хоч проект і молодий, але компанія зайняла свій ринок і осягла приблизно 170 країн світу і з нею працює 25000 співробітників. Також в Україні це один з монополістів такого виду сервісу.

2 ВИБІР АРХІТЕКТУРИ І ПЛАТФОРМИ

2.1 Основні поняття

Архітектура програмного забезпечення - розбиття системи на елементи таким чином, щоб можна було абстрактно уявити собі як частини цієї системи пов'язані між собою і хто за що відповідає, індивідуальний розвиток цих елементів і їх спосіб спілкуватися між собою. Система може бути великою і тому може бути поділена на підсистеми.[1]

У процесі циклу розроблення програмного забезпечення програмісти вирішують яка архітектура найбільш підходить їм для реалізації. Архітектура повинна спиратись на вимоги і специфіку ПЗ.

2.2 Клієнт-сервер

В цьому дипломному проекті було обрано клієнт-серверну архітектуру, оскільки вона повністю відповідає специфіці і вимогам теми цієї роботи, адже ми працюємо з клієнтами.[2]

Архітектура клієнт-сервер - одна з найбільш популярних концепцій створення веб-сайту. З самої назви зрозуміло які елементи існують у цій системі. Розглянемо їх більш детально:

- Сервер(набір серверів), задача якого - надати необхідну інформацію тому, хто робить запит до нього(клієнт);
- Клієнти - компонент системи, який звертається до сервера за допомогою сервісів;
- Мережа. Мережа слугує зв'язком між клієнтом і сервером.

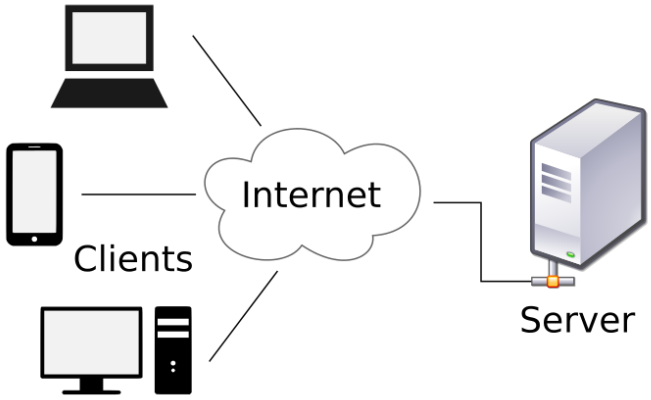


Рис. 2.1 Архітектура клієнт-сервер

2.3 Мова програмування

Для проекту була обрана мова програмування Java[3]. Java - найпопулярніша МП. Це означає, що розробники з усього світу активно підтримують, досліджують і її. Розглянемо позитивні риси:

- Практичність. Задача Java - допомогти програмістам з мінімальними зусиллями писати код.
- Сумісність. У джава роблять великий вклад такі компанії-гіганти, як Sun і Oracle. Код, написаний більш застарілою версією, все ще є придатним для користування.
- Незалежність. МП Java не залежить від платформи. Код, який був написаний одного разу буде працювати на всіх операційних системах. Коли JVM(Java Virtual Machine) компілює код, він перетворюється у байт-код, саме тому існує незалежність від ОС.
- Безкоштовність. Це є одним із головніших критеріїв організації або підприємства, адже ціна грає велику роль. Джава - абсолютно безкоштовна платформа.
- Бібліотеки. Існує безліч бібліотек, які допомагають писати компактний і швидкий код. Час - один із найважливіших

аспектів написання коду, адже компанії хочуть мати як найбільше прибутку.

- Open Source. Через те, що на Java програмують багато людей, в мережі Інтернет існують безліч бібліотек і проектів, які можна використовувати у власних цілях. У загальному вигляді платформа є вільною і відкритою для всіх.

2.4 Середовище розробки

Інтегроване середовище розробки (IDE Integrated Development Environment) - програмне забезпечення для написання коду. Складається з інструментів для написання коду, додавання плагінів, автогенерування коду, компілятора, дебагера та багато інших модулів, які допоможуть при розробці проекту[4].

Для дипломного проекту була обрана IDE Eclipse. Eclipse являється безкоштовною платформою і однією з найпопулярніших середовищ розробки програмного забезпечення. Написана мовою програмування Java. Основна ціль цієї IDE збільшення продуктивності написання коду. Eclipse підтримує кросплатформність, багато мов програмування. За допомогою цього середовища можна написати майже будь-яке клієнтське програмне забезпечення.

Eclipse IDE for Java EE Developers - середовище, у якому розробляються веб-застосунки і корпоративні проекти з використанням Java Enterprise Edition.

Архітектура Eclipse:

- Основним елементом середовища є Eclipse Runtime, у якому відтворюється код і модулі, надані цим IDE. Займається оновленнями, взаємодією с операційною системою, забезпечення допомоги;

- Другим елементом являється сама IDE. Вона відповідає за управлінням елементами програми, їх місцезнаходженням і конфігурацією, зборку проектів, пошук файлів та командну розробку.

2.5 Висновки

Проаналізувавши Java, як мову програмування, можна зробити висновок, що вона підходить для написання дипломної роботи, адже існують багато бібліотек і фреймворків, які допоможуть реалізувати архітектуру "Клєінт-сервер", якісно і швидко написати код.

3 СУЧАСНІ ТЕХНОЛОГІЇ

3.1 Основні поняття

Веб-розробка суттєво вплинула на розвинення всесвітньої павутини. Веб-розробка - це створення веб-сайту або веб-додатку[5]. Існують основні етапи для створення проекту:

- Проектування сайту або веб-додатку (збір і аналіз вимог розробка Технічного завдання, проектування Інтерфейс користувача);
- Розробка концепції сайту;
- Створення дизайн-концепції сайту;
- Створення макетів сторінок;
- Створення мультимедіа;
- Верстка сторінок і дизайнів;
- Програмування (розробка функціональних інструментів) або інтеграція в систему управління вмістом (CMS);
- Оптимізація та розміщення матеріалів сайту;
- Тестування та внесення коригувань;
- Відкриття проекту на хостингу;
- Обслуговування сайту, що працює, або його програмної основи.

У залежності від поточного завдання деякі з етапів можуть бути відсутні або бути тісно пов'язані один з іншим.

Сайт має дві основні складові: Front-end і Back-end. Front-end - візуальне представлення сайту, а back-end - робота серверу, бізнес логіка і БД.

3.2 Методи створення сайтів

Сайти можна створювати за допомогою різних методів[6]. Розглянемо технології, які на сьогоднішній день використовуються для створення веб-сайту або додатку і мають велику популярність.

3.2.1 Ручна за допомогою HTML та CSS

Hypertext Markup Language(HTML) - мова розмітки веб-сайту(не плутати з мовою програмування). HTML використовують задля створення веб-сторінок. Усі браузери вміють працювати з документами типу HTML, він сканує документ і відтворює те, що написано у документі на стороні клієнта.

Частіше за все розробник заздалегідь визначає структуру документа, тому клієнт може змінювати зовнішній вигляд, але не вміст сайту. В мові розмітки основними будівельними блоками документа є елементи HTML. Ці елементи окреслені тегами, написані за допомогою кутових дужок ("<*tag*>"). Існує багато тегів і їх можна змінювати за допомогою атрибутів. Атрибути слугують для змінення шрифту, кольору та багато інших параметрів. У свою чергу браузер вміє інтерпретувати теги і їх атрибути і знає, де буде абзац, а де форма для ведення числа, але не всі браузери інтерпретують однаково. Буває таке, що один браузер "зрозуміє" абзац, а другий - ні.

Існують два види HTML тегів:

- Теги, які слугують для змінення зовнішнього виду сторінки;
- Теги, які описують сам документ (заголовок, автор тощо)

HTML документ може бути створений у будь-якому текстовому редакторі, але задля досягнення мінімізації витрат часу на написання, у сучасному світі використовують різні редактори та конвертери, які допоможуть розробнику швидко описати документ та виправити помилки, які були допущенні.

Існує безліч редакторів, які знаходяться у мережі Інтернет і мають вільний доступ. Вибір редактора залежить від власних вподобань і зручності користування.

Cascade Style Sheets(CSS) - мова, яка описує зовнішній вигляд HTML документа. Найчастіше CSS використовують для візуалізації веб-сторінок, але цей формат також застосовується для різного виду XML-документів.

Стилі можна описати прямо у HTML, але це не є дуже гарний підхід, тому що людина, яка буде редагувати файл, буде довго змінювати CSS для кожного елемента. Рішенням цієї проблеми слугує винесення всієї CSS інформації в окремий файл, а вже потім інтегрувати його у HTML документ за допомогою спеціальних тегів:

<link rel="stylesheet" type="text/css" href="/name.css">, де name - назва файлу, у якому описані стилі, CSS таблиці. За допомогою цієї інтеграції розробник може використовувати стилі цього файлу скільки йому заманеться. Також можна комбінувати два типи використовування CSS. Тобто можна додати файл .css, а також описувати свій стиль прямо у HTML. Це залежить від вподобань розробника.

Спираючись на те, що написано вище, можна дати характеристику цьому методу розробки веб-сайтів:

- Зручність;
- Легкість написання;
- Обмежені можливості;
- Багато коду;

3.2.2 За допомогою програмних засобів

На сьогодні розробники користуються безліччю ПЗ для зручності. Це ПЗ дає змогу генерувати HTML код, проектувати у візуальному режимі веб-сайт і має великий набір функціоналу, який розробник використовує для швидкісного написання і редагування коду.

Розглянемо декілька видів систем:

- програми, що мають візуальні редактори (design-based editor) - засоби, які автоматично формують необхідний HTML-код;
- програми-редактори (code-based editors), які надають редактор і допоміжні засоби для автоматизації написання коду.

Також розглянемо design-based редактори, які використовуються програмістами для розробки:

- Adobe DreamWeaver - HTML редактор, зручний у використанні, багатий функціонал. За допомогою цього додатку можна легко створити повноцінний веб-сайт. Дає змогу працювати з HTML, CSS, JavaScript, XML;
- Microsoft FrontPage - HTML редактор, з допомогою якого можна створити веб-сторінку, сайт без знання мови HTML. Клієнт може створити сайт двома способами: з нуля або вибравши шаблон з колекції цього ПЗ. На даний момент ПЗ застаріле, зупинка підтримки сталася у 2003 році ;
- Adobe HomeSite - цей редактор має трошки інший функціонал аніж два попередники - він використовується для редагування і написання коду власноруч;
- HotDog - найперший редактор(1995 р.), має зручний інтерфейс. З плином часу додавався новий функціонал, і тому має великий функціонал. Окрім HTML підтримує CSS, VBScript, JavaScript та інші мови. Один з найбільш популярних редакторів.

3.2.3 Content Management System

Content Management System(CMS)[7] - система управління контентом. CMS дозволяє управляти текстовим і графічним контентом веб-сайту. Користувач має перед собою інтерфейс, використовуючи який, він може зберігати, публікувати інформацію, автоматизує процес передавання інформації у БД і її подальшу передачу у веб-сторінку.

CMS розрізняють за двома видами:

- Відкриті. Вони мають відкритий код для переглядання, редагування, вивчення і створення нового ПЗ на його основі(Wordpress, Drupal, Joomla);

- Закриті. Ці програми зазвичай не безкоштовні - власність підприємства або організації, також як і відкриті мають такий самий функціонал, але він набагато більший.

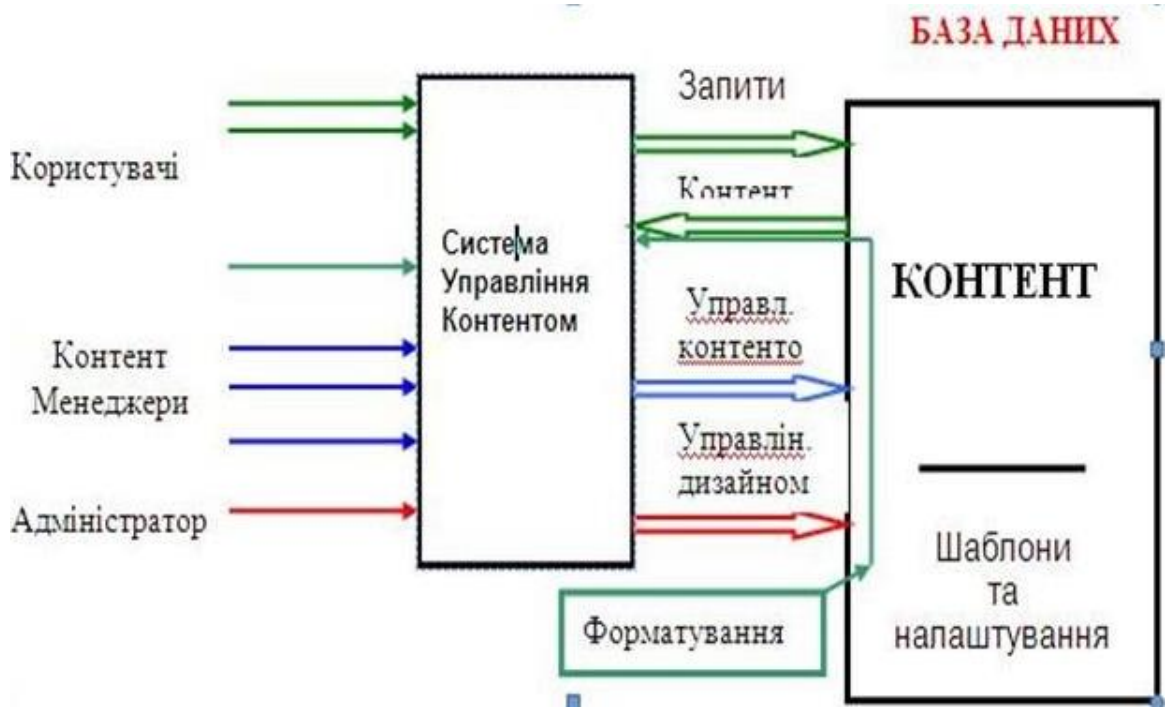


Рис. 3.1 - Система управління веб-контентом

3.3 Фреймворк Spring

Spring Framework[8] - програмний каркас, який був створений, для людей, користуючись яким, вони можуть використовувати вже написаний код для реалізації свого проекту мовою програмування Java. У Spring є безліч модулів, кожен із яких виконує свою функцію. Розглянемо основні:

- IoC-контейнер(Inversion of Control). Конфігурація компонентів і управління циклом програми.
- AoP(Aspect-Oriented Programming). Працює з функціоналом, який не може реалізувати ООП.
- Data. Реалізує роботу з реляційними базами даних, за допомогою JDBC і ORM.

- Transaction. Надання інструментів для управління транзакціями Java-об'єктів.
- MVC(Model-View-Controller). Каркас, для створення веб-застосунку, в основі якого лежить протокол HTTP і сервлети.
- Security. Модуль, який реалізує авторизацію і аутентифікацію користувача.

Для написання дипломного проекту було використано Spring IoC, Spring Data, Spring MVC, Spring Boot.

3.3.1 IoC-контейнер

IoC-контейнер[9] відповідальний за створення, зв'язок і виклику методів ініціалізації об'єктів Java. Об'єкти створені контейнером називаються "бінами"(англ. beans). Зазвичай конфігурацію описують у звичайному XML-файлі, в якому також описується конфігурація POJO(англ. Poor Old Java Object).

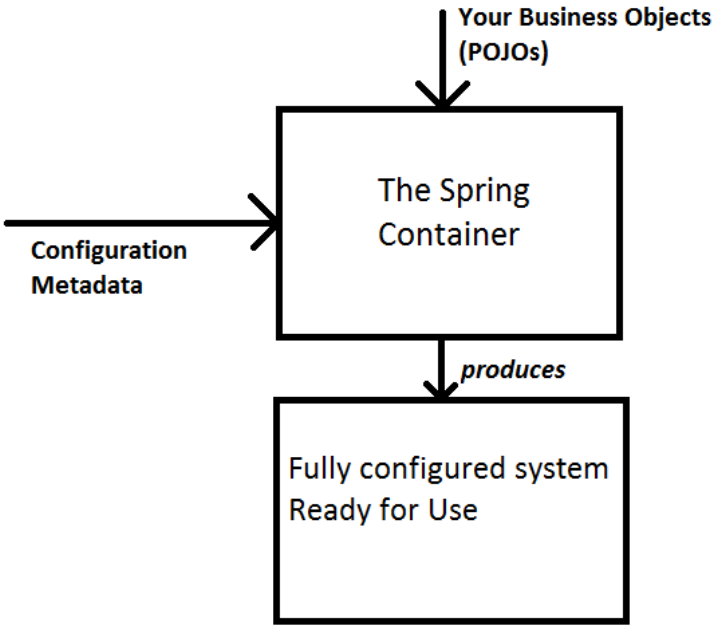


Рис. 3.2 Алгоритм роботи Spring IoC-контейнера

3.3.2 Model-View-Controller

Spring має власну платформу для створення веб-проектів. Ця платформа використовує концепцію Model-View-Controller, де Model - бізнес-модель, логіка, БД та інше, View - файл у вигляді HTML, JSP. Документ, який браузер може прочитати і зобразити у вікні браузера. Controller - клас, який відповідає на запити користувача із View і передає йому новий View, Model.

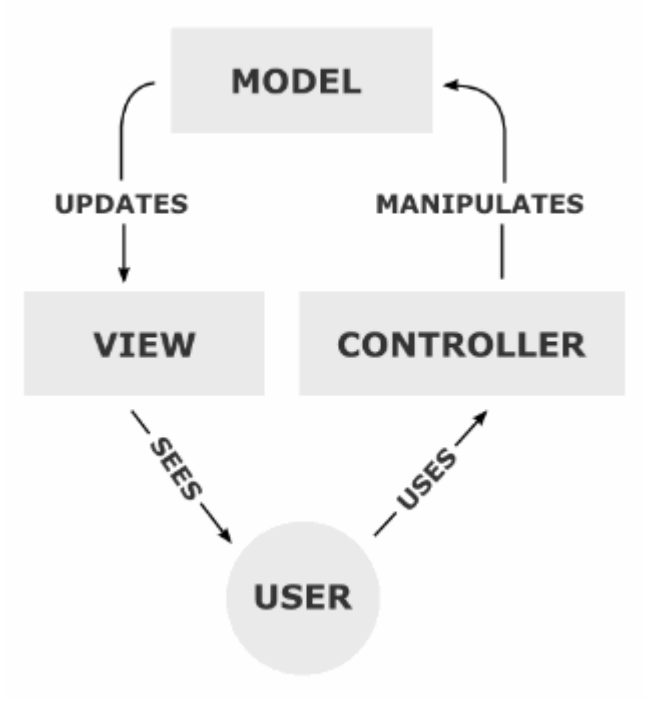


Рис. 3.3 Процес MVC

Spring MVC вміє працювати з усіма видами запитів і виконує задачу прокладання шляху до більш чистого front-end-коду. У Spring MVC є єдиний клас, який виконує функцію основного контролера, його назва - DispatcherServlet. Він відповідає за передавання управління різноманітним інтерфейсам.

Інтерфейси, які скриті від розробника і реалізують MVC у Spring:

- HandlerMapping. Шукає Java-клас, він же і контролер, у якому є метод для виконання на запит, який був створений зі сторони клієнтар;
- HandlerAdapter. Виконання методу, який підходить під запит користувача;
- Controller. Слугує мостом для View і Model. Зв'язую їх, передаючи Model у View і навпаки, із View у Model. Інтерпретує запит у відповідь;
- View. Відповідає за відповідь клієнту у виді файлу, який браузер може зобразити. Деякі запити не потребують попадати у слой моделі, а просто ідуть до контролера і назад з новим View;
- ViewResolver. Інтерфейс має колекцію образів. Обирає який саме образ буде переданим користувачу;
- HandlerInterceptor. Перехоплює і обробляє запити від користувачів;
- LocaleResolver. Отримання локальних даних, такі як мова, країна, часовий пояс.

Spring MVC надає програмісту такі можливості як: абстракцію, гарне розділення на окремі слої, змінення інтерфейсу на сучасну реалізацію.

Приклад контролеру MVC.

Лістинг:

```
@Controller
@RequestMapping(value = "/account")
public class AccountController {
private final UserRepository userRepository;
@RequestMapping(value="/info/change/submit",
method=RequestMethod.POST)
publicModelAndView ccountInfoChangeSubmit(HttpServletRequest
request, @RequestParam String telephone, @RequestParam
String address, @RequestParam String login) {
ModelAndView mav;
```

```
HttpSession session = request.getSession(true);
String user_login =
session.getAttribute("user_login").toString();
Long user_id =
Long.parseLong(session.getAttribute("user_id").toString());
if (login.equals(user_login) ||
!userRepository.isLoginExist(login)) {
userRepository.setLogin(user_id, login);
userRepository.setAddress(user_id, address);
userRepository.setTelephone(user_id, telephone);
session.setAttribute("user_login", login);
mav = new ModelAndView("redirect:/account/info");
return mav;
}
mav = new ModelAndView("user_exist");
return mav;  }
}
```

@Controller використовується для позначення класу контролером.

@RequestMapping - анотація, яка слугує для того, щоб задати методам класу-контролеру адресу, за якою вони будуть доступні на стороні клієнта. Можна використовувати також і для класу.

@RequestParam означає, що метод очікує деяких параметрів при переході на цей адрес.

3.3.3 Data

Spring Data - шар доступу до БД, підтримує всі сучасні системи управління базами даних. Data надає так званий репозиторій - це інтерфейс, який реалізований за допомогою класів-дженериків і рефлексії, надає змогу сутності і БД бути пов'язаними між собою, реалізовано за допомогою JPA(Java Persistence API).

Репозиторій надає можливість створювати методи за ім'ям, тобто нема необхідності писати запит до БД, настроювати і конфігурувати, обробляти параметри, які наприклад можуть приходити із View. Також він має вже свою власну реалізацію CRUD(Create Read Update Delete) і на сам кінець можна писати свій запит за допомогою анотації "@Query(запит)".

Приклад репозиторію.

Лістинг:

```
public interface BasketRepository extends
JpaRepository<Basket, Long>{
@Query("SELECT name FROM Basket name WHERE name.user_id =
:user_id AND name.product_name = :product_name")
Basket findByUser_IdAndProduct_name(@Param("user_id") Long
user_id, @Param("product_name") String product_name);
@Query("SELECT CASE WHEN COUNT(c) > 0 THEN true ELSE false
END "
+ "FROM Basket c WHERE c.user_id = :user_id and
c.product_name = :product_name")
boolean isExist(@Param("user_id") Long
user_id,@Param("product_name") String product_name);
@Modifying
@Query("UPDATE Basket name SET name.count = :count WHERE
name.user_id = :user_id AND name.product_name =
:product_name")
@Transactional
void increaseCount(@Param("user_id") Long user_id,
@Param("product_name") String product_name, @Param("count")
int count);

@Query("SELECT name FROM Basket name WHERE name.user_id =
:user_id")
ArrayList<Basket> findBasketsById(@Param("user_id") Long
user_id);
```

```
@Modifying
@Query("Delete FROM Basket c WHERE c.user_id = :user_id")
@Transactional
void deleteByUser_Id(@Param("user_id") Long user_id);
}
```

@Query дозволяє додати свій власний Java Persistence Query Langauge запит.

@Param вказує на те, що запит має параметр, який може набувати різного значення.

@Modifying каже про те, що вказаний метод має бути інтерпретований як модифікуючий запит.

@Transactional додає підтримку транзакцій даного методу.

3.3.4 Boot

Spring Boot допомагає авто конфігурувати проект. В ньому знаходиться вбудований Tomcat-сервер. Всі бібліотеки і плагіни пов'язуються між собою. Boot Автоматично підбирає версію цих модулів, щоб вони могли спілкуватись і функціонувати. Тобто розробнику не потрібно шукати версію того чи іншого модуля. Spring Boot сам обирає версії і слугує для як найменшого написання XML-конфігурацій. Цей Spring модуль допомогає прискорити розробку програмного забезпечення у декілька разів. Якщо програмісту потрібно якось змінити конфігурацію проекту, він може описати все що йому заманеться у анотаціях.

3.4 Hibernate

Hibernate - один із найпопулярніших фреймворків, який реалізує специфіку Java Persistence API, за допомогою ORM.

ORM(англ. Object-Relational Mapping) - технологія, яка реалізує зв'язок між БД і концепціями ОО мов програмування. Іншими словами це

відображення класу у базі даних і навпаки. Hibernate реалізує цей зв'язок за допомогою кількох анотацій: @Entity, @Id. Тобто, щоб відобразити клас у БД розробнику необхідно кілька хвилин. Це не є кінцевим функціоналом цього фреймворку: існують багато різних анотацій, які допоможуть сконфігурувати відображення, описати зовнішні й первинні ключі та багато іншого.

Для прозорішого розуміння ORM:

- Таблиця у БД = назва класу у мові програмування;
- Рядок у БД = назва атрибуту у мові програмування.

Приклад класу-сутності, який пов'язаний з БД через Hibernate.

Лістинг

```
@Entity
@Table(name="orders")
public class Order {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
Long id;
Long user_id;
String description;
String date;
String deliver_date;
public String user_telephone;
public String user_address;
String status;
Long cook_id;
Long courier_id;
}
```

@Entity помічає клас як сутність, який буде оброблятися бібліотекою.

@Table каже про те, що клас буде відображатися у таблиці. В "()" вказується назва таблиці.

@Id вказує на те, що даний атрибут буде унікальним ключем в таблиці.

@GeneratedValue використовується для того, щоб автозаповнювати значення ключа. Це можна вказати і у СУБД.

3.5 Maven

Maven - засіб для конфігурації проектів Java. Служить для додавання бібліотек і фреймворків, управління та складання проекту через XML-файл. Цей документ описує зв'язок проекту з модулями, плагінами і компонентами, які будуть використовуватись при написанні коду у застосунку.

Двигун ядра динамічно приєднує необхідні модулі до проекту, за допомогою репозиторію. Кілька рядків тексу і можна приєднати до застосунку різноманітний функціонал, ціль якого реалізувати технічне завдання. Опис модулів знаходиться у pom.xml. Цей файл генерується автоматично при роботі у IDE(англ. Integrated Development Environment) - середовище розробки.

Приклад pom.xml.

Лістинг:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.dlvr</groupId>
<artifactId>dlvr</artifactId>
<version>0.0.1-SNAPSHOT</version>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.0.3.RELEASE</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
```

```
<properties>
<project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
<java.version>1.8</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

3.6 База даних

Для проекту була обрана СУБД MySQL. MySQL найкраще підходить для малих і середніх веб-додатків. Ця СУБД надає гнучкість за рахунок підтримки різних типів таблиць. Можна створювати таблиці свого типу, а також працює на багатьох платформах і підтримує майже всі мови програмування.

СУБД є безкоштовною і відкрита для усіх користувачів у мережі Інтернет. MySQL використовується багатьма веб-додатками, що керуються базами даних, включаючи Drupal, Joomla і WordPress. MySQL також використовується багатьма популярними веб-сайтами, включаючи Facebook, Twitter, Flickr, і YouTube.

id	user_id	description	user_address	user_telephone	date	deliver_date	status
26	57	Пицца Пепперони с ...	ул. Жукова 24, 145 кв...	+38(093)388-32-65	2019-04-24 17:59:27	2019-04-24 18:59:00	cooking
27	57	Пицца Пепперони с ...	ул. Жукова 24, 145 кв...	+38(093)388-32-65	2019-04-24 18:01:36	2019-04-24 00:00:00	cooked
28	58	Пицца Пепперони с ...	ул. Жукова 24, 145 кв...	+38(093)388-32-65	2019-04-24 18:02:12	2019-04-24 19:02:00	delivering
29	60	Пицца Пепперони с ...	ул. Жукова 24, 145 кв...	+38(093)388-32-65	2019-04-26 15:50:12	2019-04-26 16:55:00	delivered
32	68	Пицца Пепперони с ...	ул. Жукова 24, 145 кв...	+38(093)388-32-65	2019-05-20 17:41:58	2019-05-20 18:59:00	ordered
33	32	Пицца Пепперони с ...	ул. Жукова 24, 145 кв	+38(093)388-32-63	2019-05-20 18:47:14	2019-05-20 19:47:14	cooking
34	32	Пицца Пепперони с ...	ул. Жукова 24, 145 кв	+38(093)388-32-63	2019-05-20 18:48:17	2019-05-20 19:48:17	delivered
35	69	Бургер Эль классик...	ул. Жукова 24, 145 кв...	+38(093)388-32-65	2019-05-21 10:20:45	2019-05-21 12:01:00	delivered
36	70	Пицца Пепперони с ...	ул. Жукова 24, 145 кв...	+38(093)388-32-65	2019-05-22 16:58:11	2019-05-22 17:59:00	ordered

Рис 3.4 Вигляд таблиці у MySQL

3.7 JSP

JavaServer Pages (JSP) - це технологія, яка допомагає розробникам програмного забезпечення створювати динамічно створювані веб-сторінки на основі HTML, XML або інших типів документів. Випущений в 1999 році компанією Sun Microsystems, JSP схожий на PHP і ASP, але він використовує мову програмування Java.[10]

Для розгортання та запуску JavaServer Pages потрібний сумісний веб-сервер з контейнером сервлетів, таким як Apache Tomcat або Jetty.

У проекті реалізована бібліотека JSTL, яка слугує для розширення технології JSP. Її задача полягає у рішення звичайних завдань, таких як обробка даних XML, умовне виконання, доступ до бази даних, цикли та інтернаціоналізація. Найголовнішим у веб-сайті було використання циклів. Вони використовувалися для відображення у JSP даних, які були передані на стороні сервера.

Приклад використання тега "c", циклу, у JSP.

Лістинг:

```
<form action="/account/courier_page/submit_order" method =
POST style="width: 770px;">
<h2>Выберите заказ, если вы уже доставили его и нажмите
"Готово"</h2>
<c:forEach var="orderlist" items="${orderlist}">
<p><input name="order_id" type="radio"
value="${orderlist.id}">Id заказа:${orderlist.id }
<br>
${orderlist.description }. Дата доставки:
${orderlist.deliver_date }
</p>
<br>
</c:forEach>
<input type="submit" value="Готово">
</form>
```

3.8 HTTP

Протокол передачі гіпертексту (HTTP) є протоколом прикладних програм для розподілених, спільних, гіпермедійних інформаційних систем. HTTP - це основа передачі даних для World Wide Web, де гіпертекстові документи включають гіперпосилання на інші ресурси, які користувач може легко отримати, наприклад, натисканням миші або натисканням на екран у веб-браузері. HTTP був розроблений для полегшення гіпертексту.

У розробці частіше за все використовуються тільки два методи запиту клієнта до сервера: GET і POST.

3.8.1 GET запит

Метод GET запитує представлення зазначеного ресурсу. Запити, які використовують GET, повинні тільки отримувати дані і не повинні мати іншого ефекту. W3C опублікував керівні принципи щодо цього розрізнення, сказавши: "Дизайн веб-додатків повинен бути проінформований за вищезазначеними принципами, а також за відповідними обмеженнями".

3.8.2 POST запит

Метод POST вимагає, щоб сервер отримав об'єкт, який знаходиться у запиті, як новий підлеглий веб-ресурс, ідентифікований адресом URI. Дані POSTed можуть бути, наприклад, анотацією для існуючих ресурсів; повідомлення для дошки оголошень, групи новин, списку розсилки або теми коментарів; блок даних, який є результатом подання веб-форми в процес обробки даних; або елемент для додавання до бази даних.

3.9 MessageDigest5

Паролі користувачів зберігаються у базі даних за допомогою хешування. Це робиться для того, щоб хакер чи злодій який отримав доступ до БД не зміг пройти аутентифікацію.

MD5 (Message Digest 5) - 128-бітний алгоритм хешування. Його задача зашифрувати дані, створити "відбиток" від будь-якої величини повідомлення.

На вхід алгоритму надходить вхідний потік даних, хеш якого необхідно знайти. Довжина повідомлення вимірюється в бітах і може бути будь-який (в тому числі нульовою). Запишемо довжину повідомлення в L. Це число ціле і невід'ємне. Кратність будь-яким числам необов'язкова. Після надходження даних йде процес підготовки потоку до обчислень.

Хеш має 128 бітів, 16 байтів і зазвичай має вигляд 32-х шістнадцятирічних чисел.

Приклад хешування строки "md5":

MD5("md5") = 1BC29B36F623BA82AAF6724FD3B16718

3.10 Висновки

Для проекту було обрано сучасні технології, якими користуються багато IT-компаній і розробників, такі як IBM, Google, RedHat, JetBrains.

- Spring Framework;
- Hibernate (ORM);
- JSP і бібліотека JSTL;
- Maven;
- СУБД MySQL;
- MD5;
- HTTP.

Цих технологій буде достатньо для створення проекту, який задовольнить мету цієї дипломної роботи і для подальшого розвитку цієї системи.

4 ПРОЕКТУВАННЯ СИСТЕМИ

4.1 Unified Modeling Language

UML(Unified Modeling Language) - мова графічного опису моделі у сферах розробки програмного забезпечення, бізнес-процесів, проектування систем і відображення структури. Не є мовою програмування, але за допомогою цієї мови можна генерувати код.

UML - стандарт, яким користуються для абстрактного моделювання системи, так звана UML-модель.

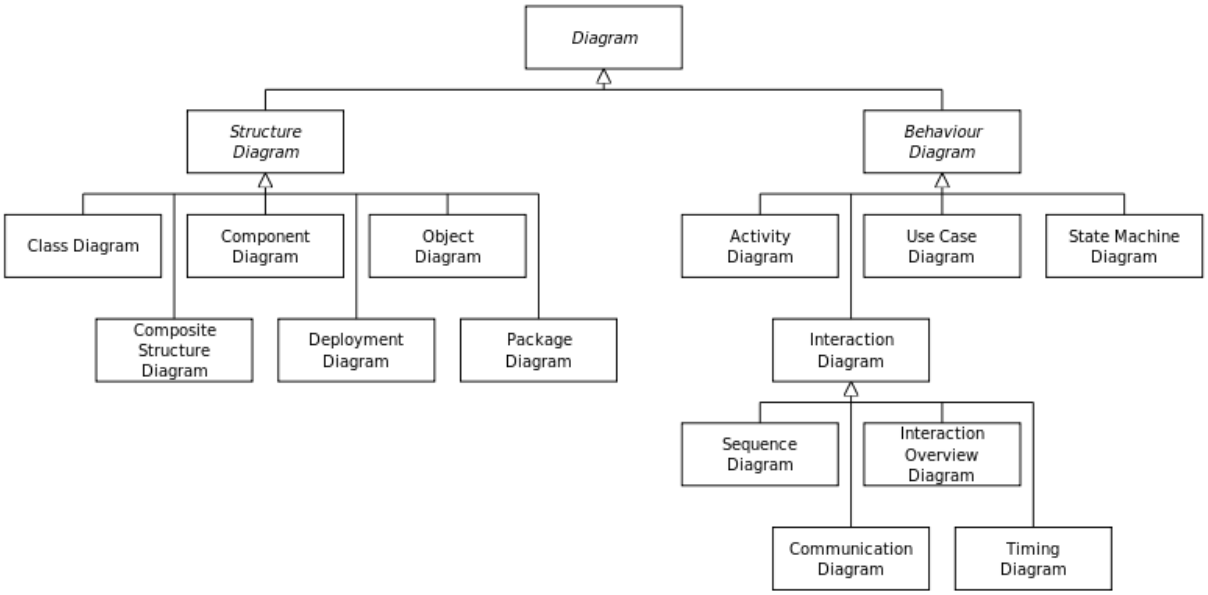


Рис 4.1 Ієрархія UML діаграм

Діаграма - графічне представлення даних лінійними відрізками або геометричними фігурами, що дозволяє швидко оцінити співвідношення декількох величин. Являє собою геометричне символьне зображення інформації із застосуванням різних прийомів техніки візуалізації.

4.2 Функціональна модель

Діаграма варіантів використання(Use Case Diagram) - діаграма в UML, яка зображає зв'язок між акторами і прецедентами[11]. Ця діаграма являється

графом. Сенс полягає у тому, безліч акторів взаємодіють із системою. Система у даному випадку - якась кількість варіантів використання, які повинні бути пов'язані змістом.

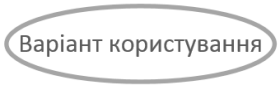
Розширення - відношення між прецедентами між базовим варіантом і його спеціальним випадком.

Включення вказує на те, що варіант використання може бути здійснене тільки при виконанні базового варіанту.

Узагальнення існує, щоб зобразити відповідну спільність ролей. Схоже на наслідування у МП.

Асоціація показує, що актор може вказувати на те, що він ініціює відповідний варіант використання

Актор - сутність, яка впливає на систему. Також може виступати у ролі людини, системи, підсистеми. Він робить певні дії над системою і взаємодіє з нею.



Прецедент - специфікація послідовності дій, які може здійснювати система, підсистема або клас, взаємодіючи з акторами.

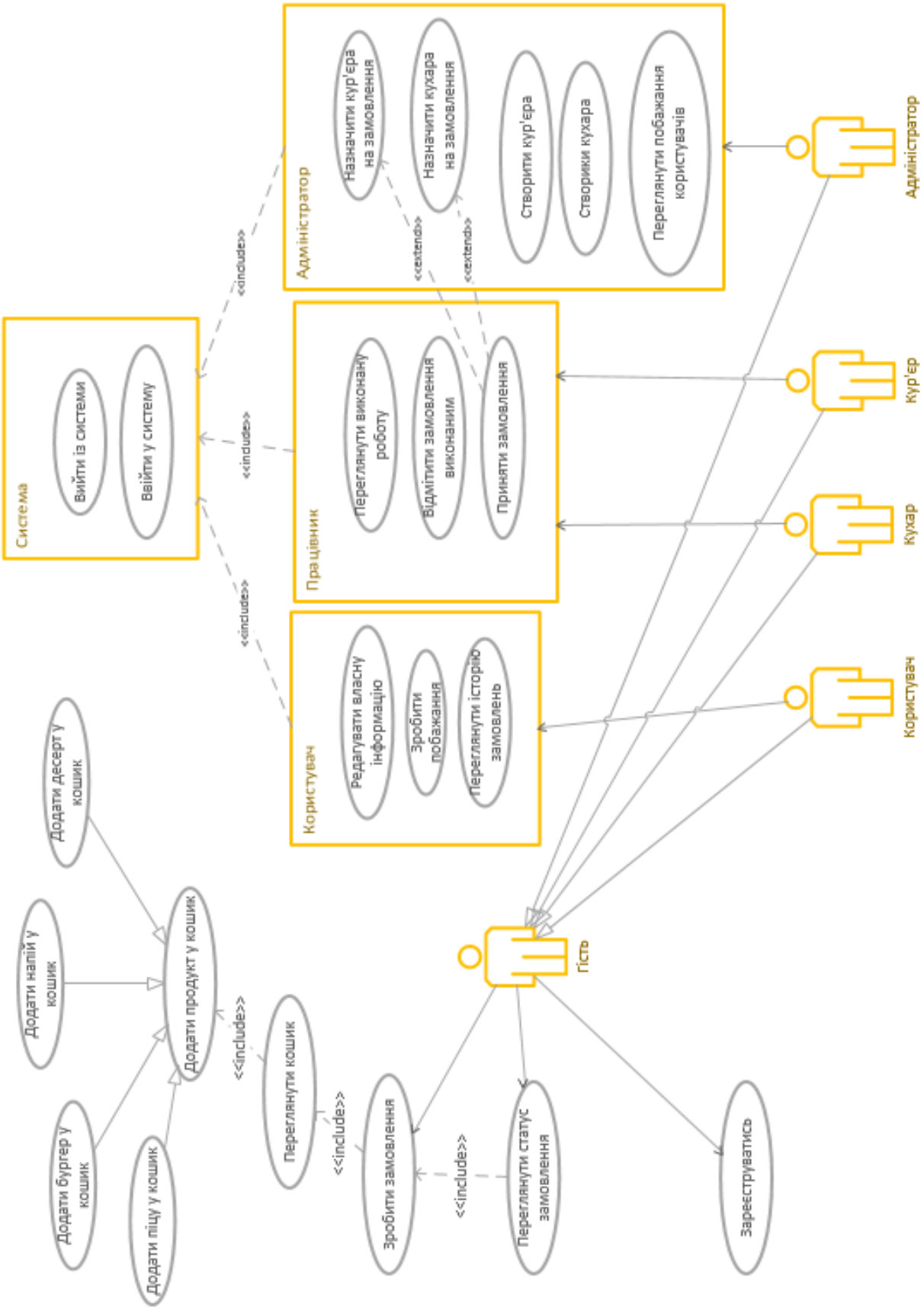


Рис 4.2 Use Case Diagram

Користувач, кухар, кур'єр і адміністратор унаслідуються від гостя і вміють робити все те саме, що і він. Розглянемо, що саме вміють актори:

Гість:

- *Додати продукт у кошик*. Увійти на одну з чотирьох сторінок з продуктами і додає їх у свій кошик
- *Переглянути кошик.* Увійти на сторінку з переглядання свого кошика. Може бути здійснене, тільки якщо кошик не є порожнім.
- *Зробити замовлення.* При перегляданні кошика користувач має змогу замовити те, що він додав.
- *Переглянути статус замовлення.* Перейти на сторінку, щоб слідкувати за своїм замовленням.
- *Зареєструватися.* Гість також має змогу зареєструватися.

Користувач:

- *Редагувати власну інформацію.* Користувач може змінити інформацію, яку він вводив при реєструванні.
- *Зробити побажання.* Залишити коментар, щоб адміністратор міг прочитати і внести якійсь поправки.
- *Переглянути історію замовлень.* Клієнт переглядає історію замовлень, які він зробив за увесь час.

Кухар:

- *Переглянути виконану роботу.* Кухар дивиться, що він приготував за весь період праці.
- *Прийняти замовлення.* Кухар помічає замовлення, як замовлення, яке "готується".
- *Відмітити замовлення виконаним.* Кухар помічає замовлення "приготованим".

Кур'єр:

- *Переглянути виконану роботу.* Кур'єр дивиться, що він доставив за весь період праці.

- *Прийняти замовлення.* Кур'єр помічає замовлення, як замовлення, яке "доставляється".
- *Відмітити замовлення виконаним.* Кухар помічає замовлення "доставленим".

Адміністратор:

- *Переглянути побажання користувачів.* Перегляд побажань користувачів для подальших змінень сервісу.
- *Створити кухаря.* Заповнення інформації і реєстрація нового кухаря.
- *Створити кур'єра.* Заповнення інформації і реєстрація нового кур'єра.
- *Назначити кухаря на замовлення.* Адміністратор назначає кухаря, щоб він почав готувати.
- *Назначити кур'єра на замовлення.* Адміністратор назначає кур'єра, щоб він доставив замовлення.

Як видно з діаграми, весь цей функціонал можна виконувати, тільки якщо користувач сайту ввійшов у систему: пройшов авторизацію і аутентифікацію.

4.3 Концептуальна модель

Діаграма класів - представлення моделі системи у статичному вигляді. На цій діаграмі зображено такі елементи як: класи, інтерфейси, атрибути, їх відношення між собою та методи.[12]

При проектуванні, розробник повинен відобразити їх поведінку, стан і взаємозв'язок. На кожному етапі здійснюється абстрагування від мальовничих деталей і концепцій, які не відносяться до реальності.

Розглянемо елементи, які слугують для позначення зв'язку у класовій діаграмі:

Рис. 4.3 Види зв'язку

Асоціація показує зв'язок між двома об'єктами таким чином, що з однієї сутності можна перейти до другої, іншими словами один об'єкт зберігає у собі інший і навпаки.

Успадкування вказує на те, що клас "дитина" має всі ті самі атрибути і методи, що й спадок. Цей механізм реалізує поліморфізм.

Реалізація/Імплемантація відображає те, що клас реалізує якогось роду інтерфейс.

Залежність вказує, що один клас є абсолютно залежним від іншого і будь-які зміни одного класу відобразяться на базовому, але не навпаки.

Агрегація - коли об'єкт одного класу має колекцію із об'єктів іншого. Агрегація є "слабким" зв'язком, тобто якщо не буде існувати один об'єкт це не означає, що не буде існувати інший. Приклад: вкрали меблі з квартири, але вона все ще є квартирою.

Композиція = агрегація, але має сенс більш "жорстокого" зв'язку. Приклад: кімната без квартири не існує, і тому тут буде використовуватися композиція.

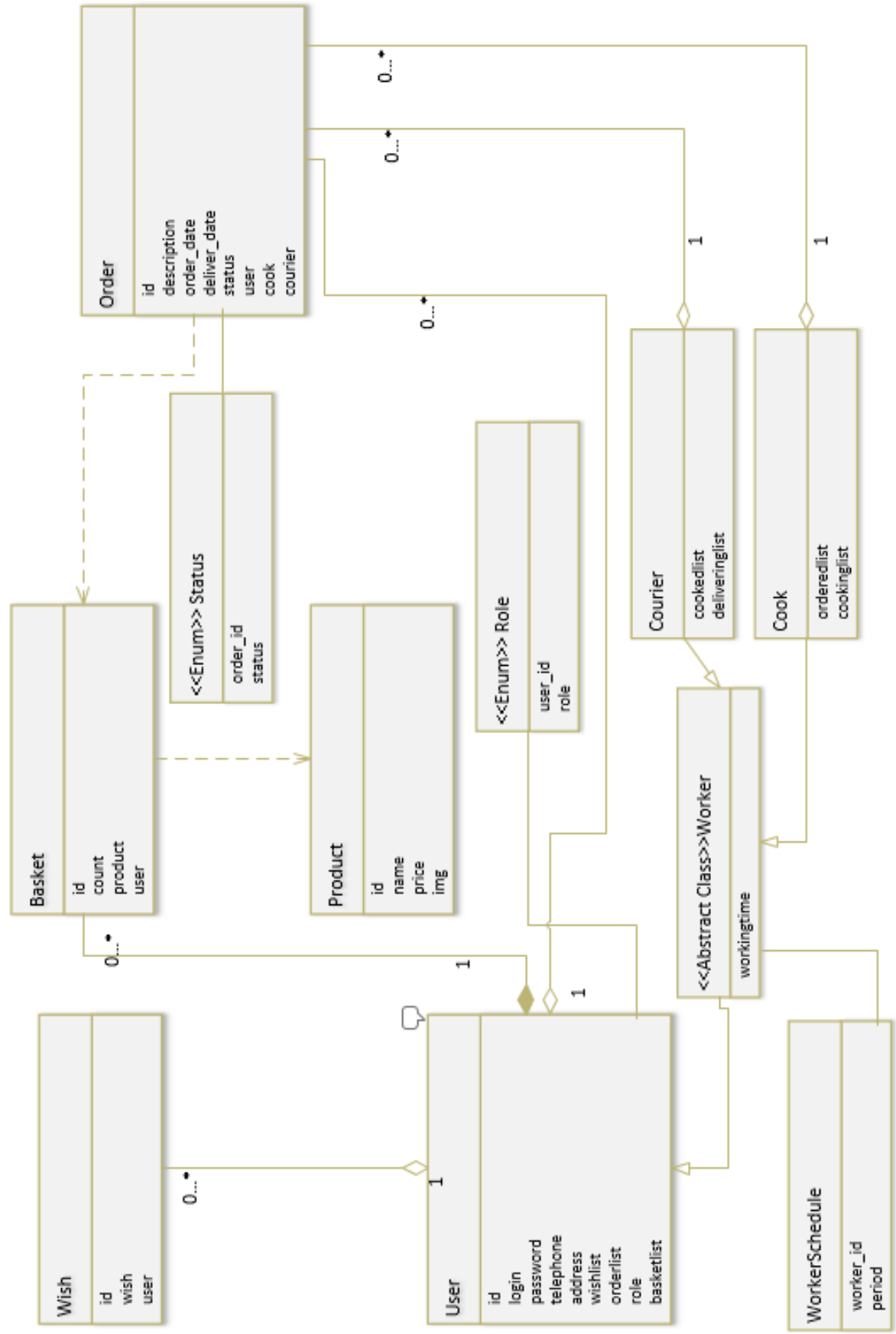


Рис. 4.5 Класова діаграма

Нотація	Пояснення	Приклад
0...1	Нуль або один екземпляр	Кіт має або не має хазяїна
1	Обов'язково один екземпляр	У кота одна мама
0...* або *	Нуль або більше екземплярів	У кота можуть бути, а можуть і не бути діти
1...*	Один або більше екземплярів	У кота є хоча б одне місце, де він спить

Табл. 1 Зв'язки, які можуть бути між сутностями

Розглянемо взаємозв'язки у діаграмі:

- User (1) - (0...*) Wish. Зв'язок - агрегація. Побажання, які зробив користувач повинні залишатись. Один користувач може залишити безліч побажань, а може й не залишити;
- User (1) - (0...*) Basket.У системі кошик реалізовано так, що кошик, який бачить користувач програмно складається із багатьох кошиків(Basket). Тому зв'язок - композиція: кошик не може існувати без користувача. Один User має багато Baskets, чи не має взагалі, тобто ще не додав продукт у кошик;
- User (1) - (0...*) Order. Зв'язок - агрегація. Клієнт який тільки зареєструвався не має замовлень, але у замовлення(Order) є тільки один клієнт;
- У кожного User'а є своя роль. Їх всього п'ять: гість, користувач, кухар, кур'єр, адміністратор. Ці ролі зберігаються у переліку Enum Role.

- Абстрактний клас Worker успадковуються від класу User, щоб виділити клас людей які працюють у системі: у них є атрибут workingtime, який позначає час роботи працівника;
- Кур'єр і кухар успадковуються від абстрактного класу Worker;
- Cook (1) - (0...*) Order. Зв'язок - агрегація. У одного кухаря може бути багато замовлень, які він готує чи приготував. Може й не бути жодного замовлення;
- Courier (1) - (0...*) Order. Зв'язок - агрегація. У одного кур'єра може бути багато замовлень, які він доставляє чи доставив. Може й не бути жодного замовлення;
- Basket залежить від Product. Клас Product - меню. При зміні атрибутів продукту буде змінюватись кошик.
- Order залежить від Basket. Замовлення формується на базі кошику;
- У замовлення (Order) є свій статус замовлення, їх п'ять: замовлено, готується, приготовлене, доставляється, доставлено. Ці статуси зберігаються у таблиці-переліку під назвою Enum Status.

Дивлячись на класову діаграму можна уявити, як представлені класи у мові програмування.

4.4 Загальний опис роботи системи

Діаграма діяльності (англ. activity diagram) — графічне представлення робочих процесів поетапної діяльності та дій з підтримкою вибору, ітерації та паралельності. У UML діаграми діяльності призначені для моделювання як обчислювальних, так і організаційних процесів (тобто робочих процесів), а також потоків даних, що перетинаються з відповідними діями. Хоча діаграми активності в першу чергу показують загальний потік контролю, вони також можуть включати елементи, що показують потік даних.[13]

Дія (англ. action) є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграми активності будуються з обмеженої кількості фігур, з'єднаних стрілочками. Найважливіші типи фігур:

- прямокутники з округленими кутами позначають дії;
- ромби позначають рішення;
- риски позначають початок (розподіл) чи кінець (об'єднання) паралельних активностей;
- чорний кружок позначає старт (початковий стан) процесу;
- чорний кружок в колі позначає кінець (кінцевий стан).

Стрілки ведуть від старту до кінця і позначають порядок в якому відбуваються активності. Діаграма активностей може вважатись формою блок-схеми.

4.4.1 Загальна діаграма діяльності

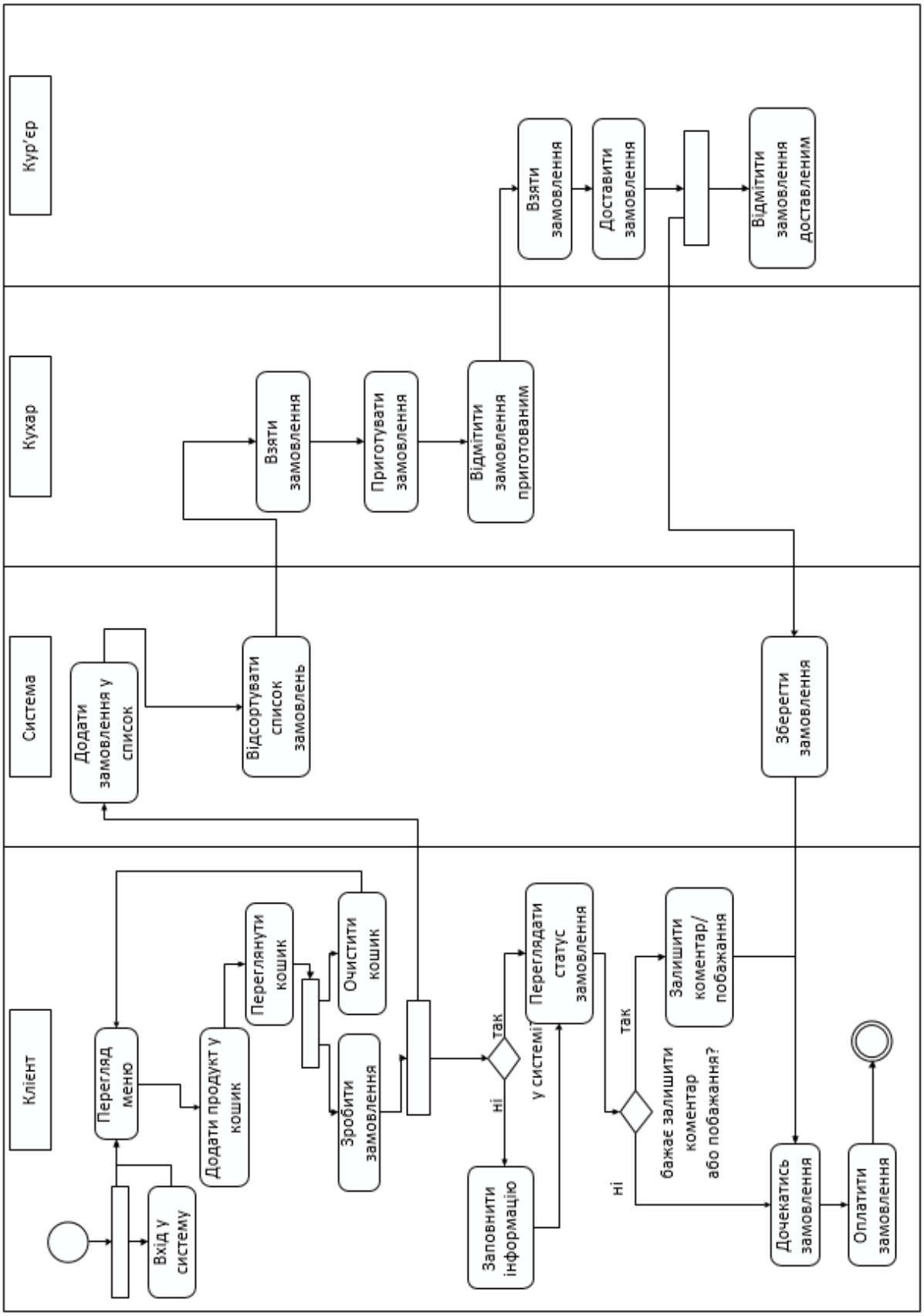


Рис 4.6 Загальна діаграма діяльності

Опис діаграми зі сторони клієнта:

Клієнт може бути гостем(Unknown) і користувачем(User), щоб стати користувачем, йому потрібно зробити *вхід у систему*. Перейти на сторінки з продуктами, щоб *переглянути меню* та *додати* те, що потрібно.

Потім, щоб *зробити замовлення,* користувачу необхідно *переглянути кошик*. Якщо клієнт передумав або вирішив змінити щось у замовленні, він може *очистити кошик* і перейти до пункту *переглянути меню.*

Якщо користувач у системі, то він може замовити без інформації, бо вся інформація, необхідна для замовлення, вже знаходиться у системі, с тих пір коли він увійшов у систему, а якщо ні, то клієнт буде переведений на сторінку бланку, щоб він *заповнив інформацію.* Після того, як користувач зробив замовлення, можна *переглядати статус замовлення*.

По бажанню, можна *залишити коментар або побажання*. *Дочекатись* замовлення і *оплатити* його.

Опис діаграми зі сторони системи:

Коли відбувається замовлення, система додає це замовлення у БД і передає у *відповідні списки* клієнтам, кухарям, кур'єрам і адміністраторам. По запиту користувачів *сортує* їх так, щоб їм було зручно переглядати інформацію.

По завершенню доставки система *зберігає* всю інформацію про замовлення у БД для подальшого аналізу і статистики.

Опис діаграми зі сторони кухаря:

Кухар в свою чергу має отримати список замовлень і *взяти* його на приготування. Коли він відмічає, що замовлення готується, то воно переноситься у другий список, який має статус - "готується". На той час, як він *приготував замовлення*, йому треба *відмітити*, що замовлення приготоване.

Опис діаграми зі сторони кур'єра:

Кур'єр має список приготованих замовлень, які можна доставити. Він *бере* його і доставляє у потрібне йому місце. Коли клієнт оплатив замовлення, кур'єр *відмічає* його у своїй системі.

4.4.2 Діаграма діяльності адміністратора

Рис. 4.7 Діаграма діяльності адміністратора

Опис діаграми:

Адміністратор може *створити* нового кухаря або кур'єра, наприклад якщо на роботі з'явився новий працівник.

Якщо адміністратору потрібно спеціально *назначити* на відповідне замовлення працівників, то він *назначає кухаря,* потім слідкує за тим, щоб замовлення приготувалось і *назначає кур'єра*.

4.5 Карта сайту

Карта сайту - графічне зображення веб-сайту. Карта зображає список сторінок для пошукових систем або користувачів. Має ієрархічний тип вигляду. Використовується для навігації і показує взаємозв'язки між сторінками веб-сайту. Якщо потрібно відстежувати зміни у розробці, а також відновляти і конфігурувати гіперпосилання, то карта сайту необхідна для виконання цих задач.

Деякі гіперпосилання позначені красним хрестиком. Це може означати, що адрес не знайшовся, в доступі відмовлено або потрібен пароль чи параметри. У випадку проекту ці хрестики означають, що потрібні параметри і паролі, тобто ви не можете просто так перейти по даному гіперпосиланню.

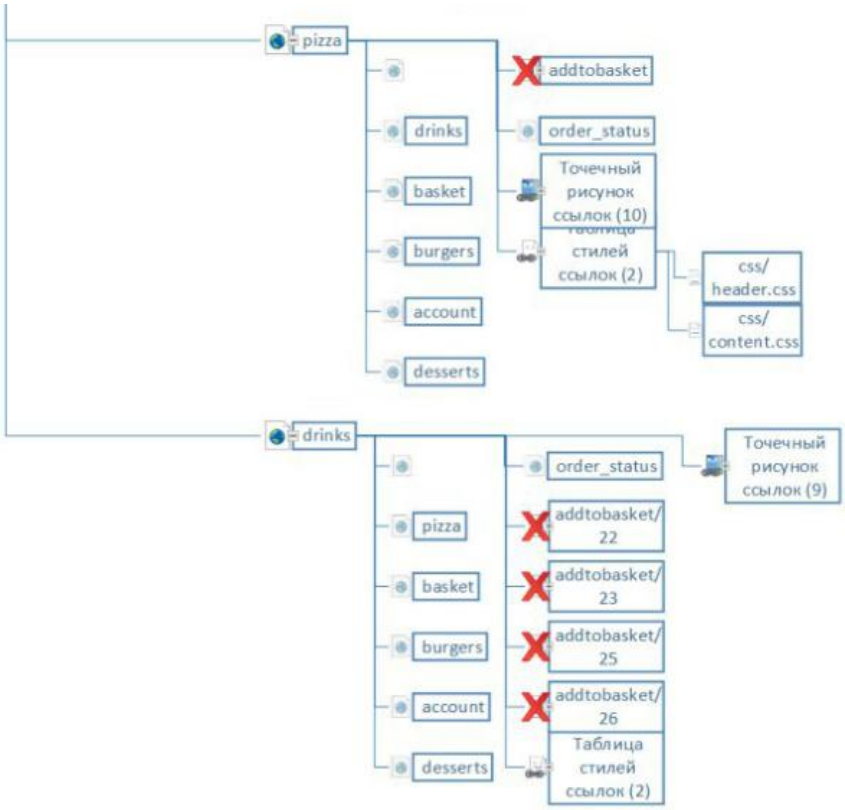


Рис. 4.8 Карта сайту, перша частина

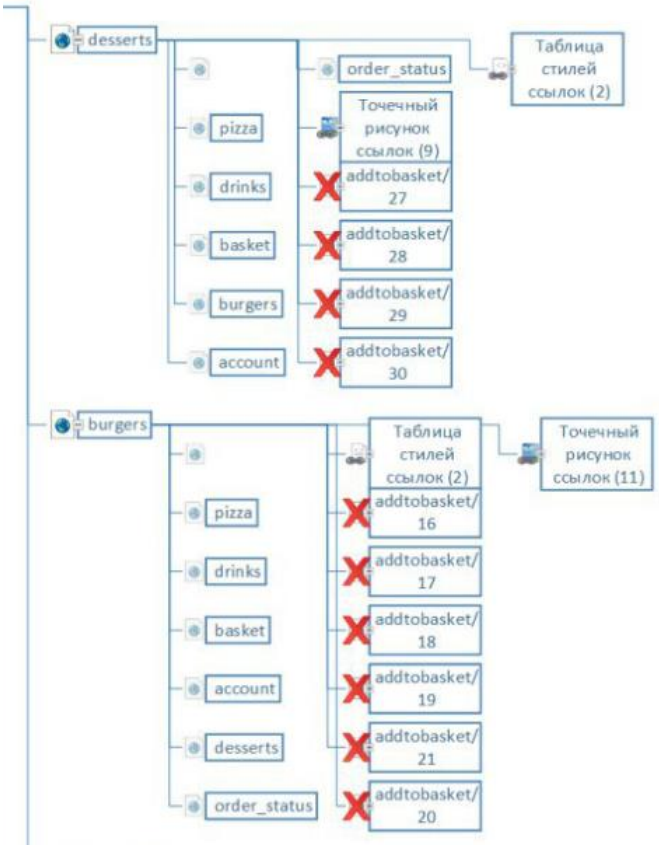


Рис. 4.9 Карта сайту, друга частина

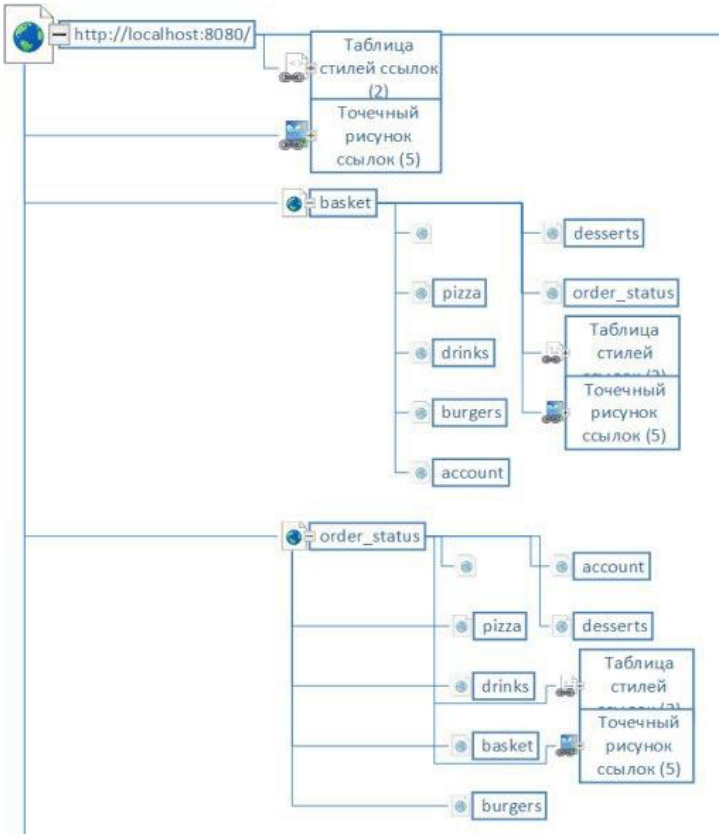


Рис. 4.10 Карта сайту, третя частина

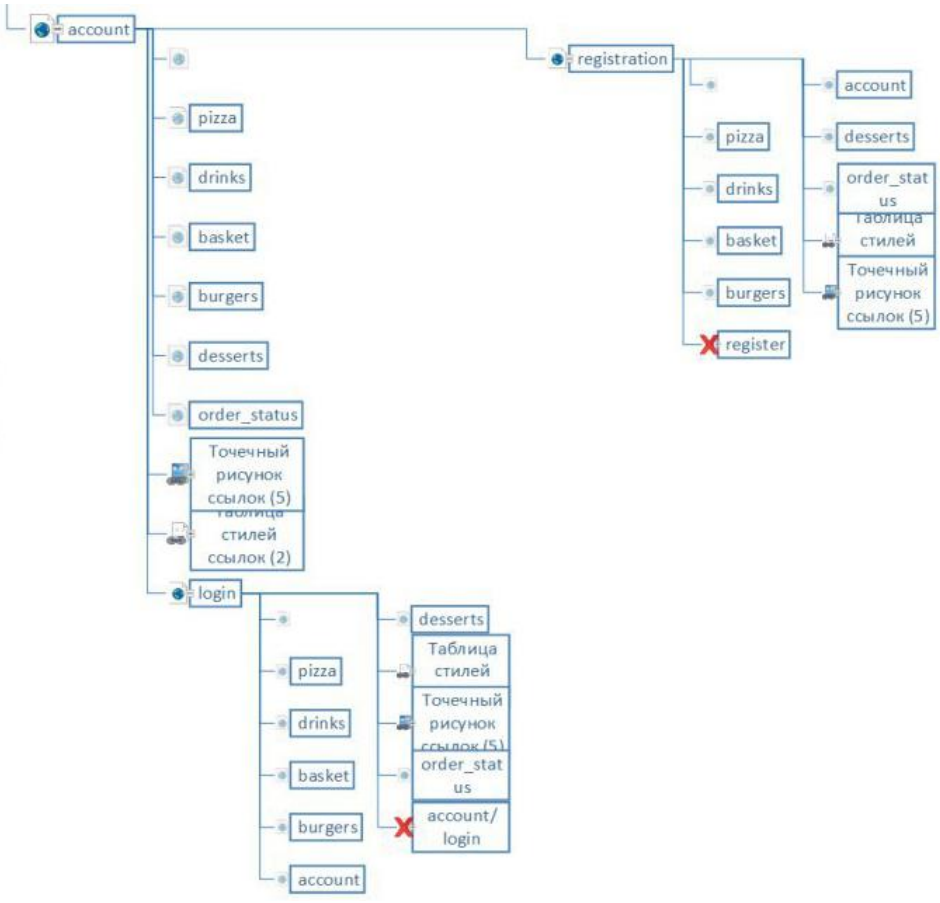


Рис. 4.10 Карта сайту, четверта частина

4.6 Висновки

Було спроектовано модель системи за допомогою таких UML діаграм як: Use Case, Activity, Class і карти сайту. В цих моделях описано всі елементи системи і їх зв'язки між собою.

5 ДЕМОНСТРАЦІЯ РОБОТИ СИСТЕМИ

Демонстрацію проекту можна побачити у браузері за адресою "*localhost:8080"*. Сервером було обрано Apache Tomcat. Все, що потрібно зробити - запустити програмне забезпечення за допомогою IDE.

Як вже відомо з діаграм у системі є п'ять ролей: гість, клієнт, кухар, кур'єр, адміністратор. Користувач сайту "спілкується" з сервером за допомогою GET і POST запитів. Розглянемо роботу кожного з них, щоб мати уявлення: хто має який функціонал і хто за що відповідальний.

5.1 Гість

Задача гостя веб-застосунку - замовити їжу. Далі буде продемонстровано як це робиться у браузері:

Для початку треба зайти на сайт за адресою "localhost:8080". У вікні браузера ви побачите наступне:

Рис 5.1 Початкова сторінка

Для додавання продуктів у кошик перейдіть на одне з чотирьох посилань: "Пицца", "Бургеры", "Напитки", "Десерты" та додайте все, що забажаєте за допомогою натискання кнопки "В Коризну".

На рисунку 5.2 проілюстровано як додати піцу у кошик.

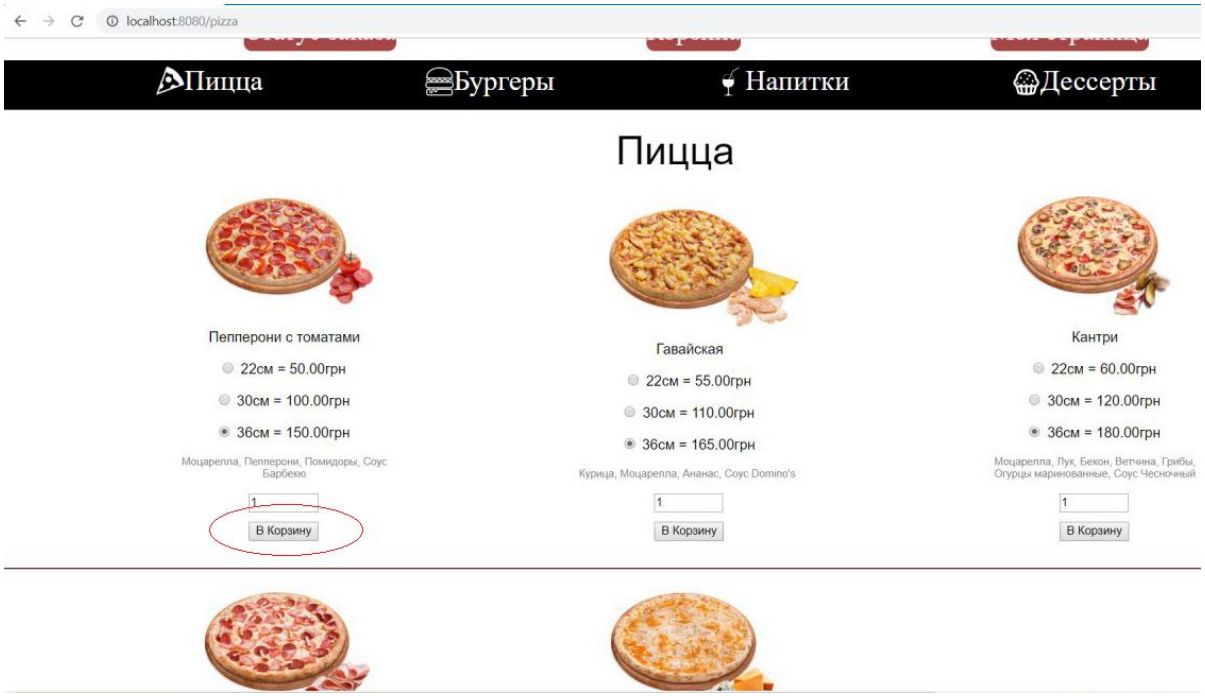


Рис. 5.2 Сторінка з піцою

Коли все, що потрібно додано у кошик треба перейти на сторінку переглядання кошику, за допомогою натискання "Корзина". Якщо користувач нічого не додав, то йому буде зображено повідомлення "Вы еще ничего не заказали!".

На рисунку 5.3 зображено кошик, у який додало три піци "Пеперoні".

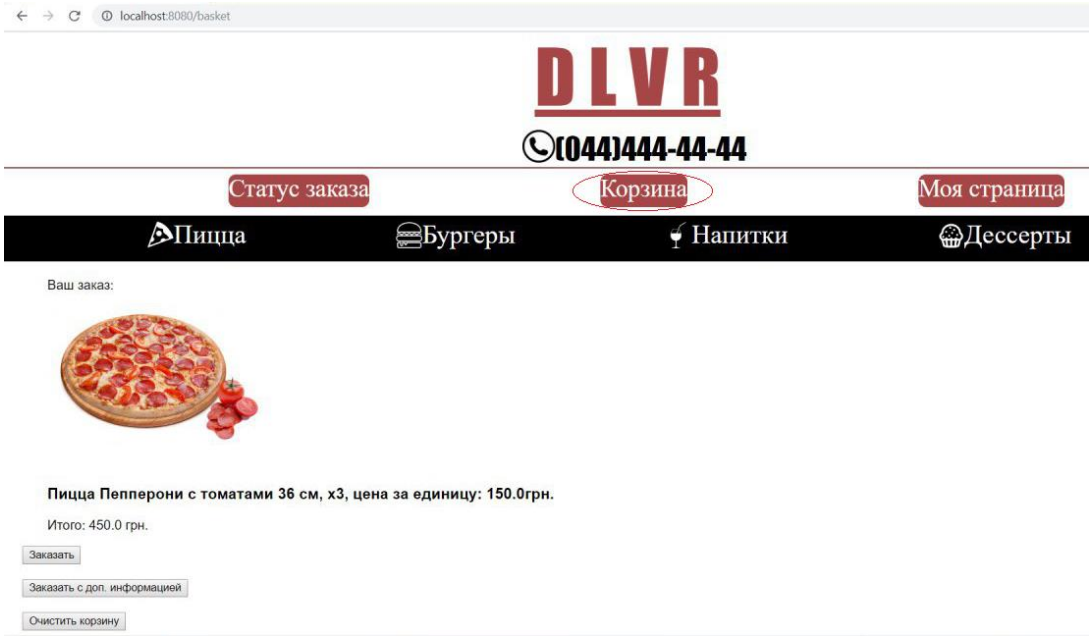


Рис. 5.3 Сторінка переглядання кошика

У гостя є три альтернативи "Заказать", "Заказать с доп. информацией" і "Очистить корзину". В випадку гостя "Заказать" буде мати такий самий сенс як і "Заказать с доп. информацией", тому що гість ще не надавав своєї контактної інформації, тому якщо він натисне кнопку "Заказать", то він буде напрямлений до "Заказать с доп. информацией", щоб заповнити інформацію для замовлення їжі.

На рисунку 5.4 зображено бланк, який треба заповнити і натиснути кнопку "Заказать" для остаточного замовлення.

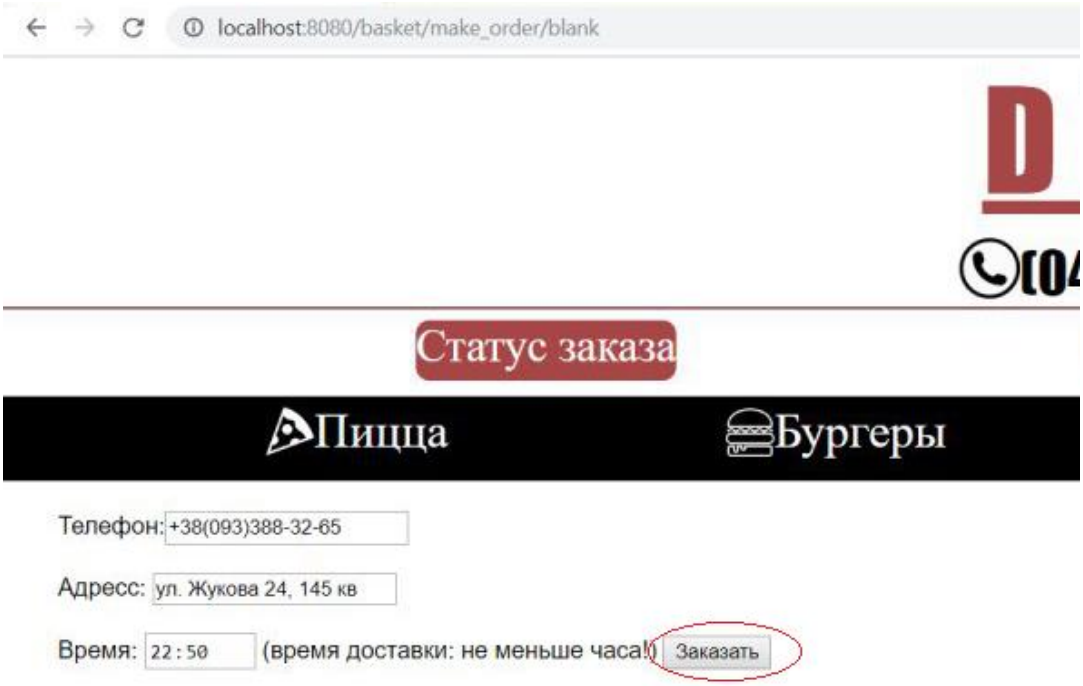


Рис. 5.4 Бланк заповнення інформації

У цьому бланку є три поля, які повинні бути заповненні: Телефон, адреса і час доставки.

Телефон має валідацію на українські номера, тобто можна ввести тільки номер, який починається з "+380".

Адреса не має жорстких обмежень, але слід ввести найбільш точну інформацію.

На час є валідація у вигляді теперішнього часу плюс одна година, тобто замовлення можна зробити тільки на 14:00, якщо у момент замовлення час=13:00.

Коли бланк заповнений, потрібно нажати кнопку "Заказать". Одразу після натискання, при успішному замовленні, у сторінці браузера буде привітання з замовленням.

Потім гість може переглядати свої замовлення, натиснувши "Статус заказа". Якщо гість нічого не замовив, то він буде мати повідомлення у вигляді "Вы еще ничего не заказали!".

На рисунку 5.5 зображено сторінку для перегляданння статусу замовлення.

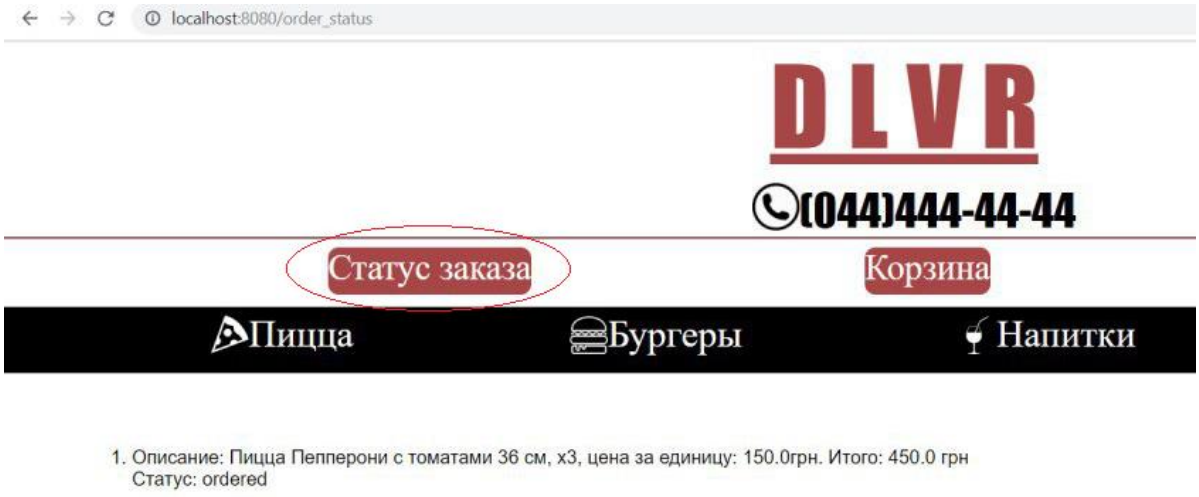


Рис. 5.5 Сторінка статусу замовлень

Оскільки було зроблено тільки одне замовлення, ми бачимо короткий опис єдиного замовлення і його статус, замовлення має п'ять статусів:

- ordered. Замовлення тільки було замовлене - ще не готується;
- cooking. Готується кухарем;
- cooked. Приготовлене;
- delivering. Кур'єр доставляє;
- delivered. Кур'єр доставив замовлення.

Також гість має спромогу зареєструватися, щоб стати користувачем(User). Для цього йому потрібно перейти у "Моя страница" і обрати "зарегистрируйтесь".

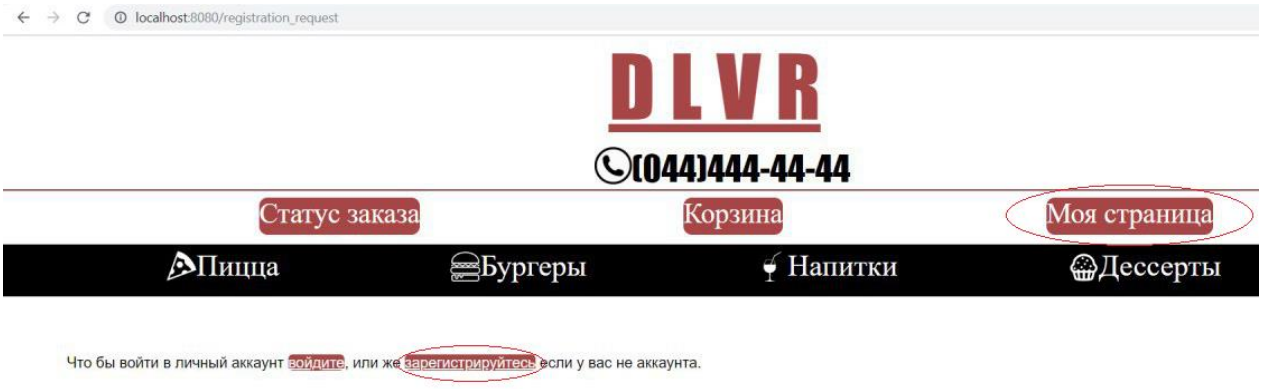


Рис 5.6 Шлях до реєстрації

Потім гість перейде на сторінку з реєстраційним бланком, який йому буде потрібно заповнити, щоб зареєструватися.

На рисунку 5.7 зображено як виглядає реєстраційна сторінка.

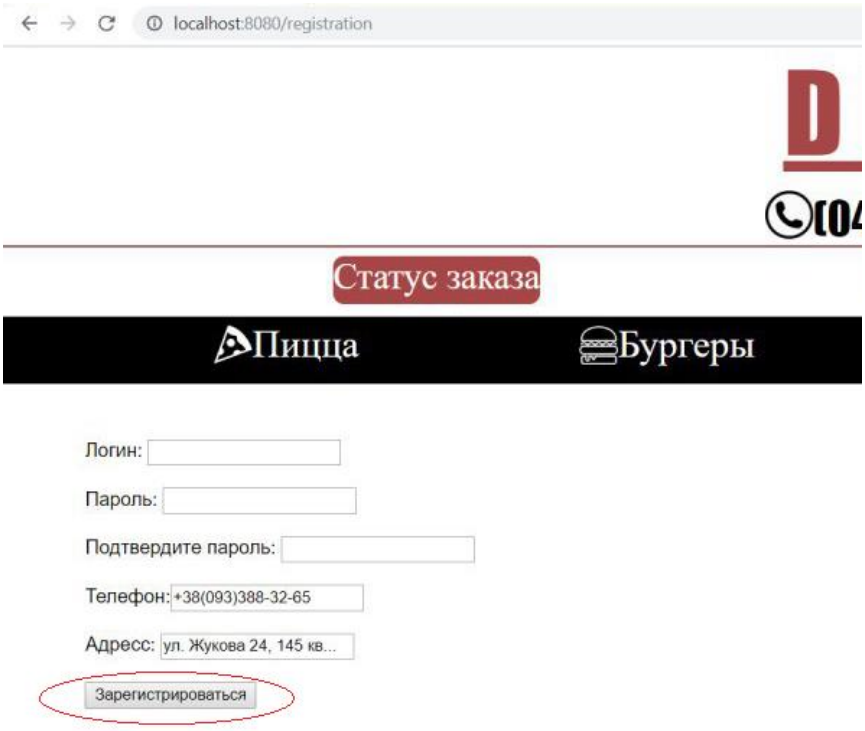


Рис 5.7 Реєстраційний бланк

Логін має валідацію на латинські букви, цифри і такий символ як "_". Довжина логіна має бути від 3 до 15 символів. На пароль накладена та сама валідація, що і на логін. Поле "Подтвердите пароль" має бути таким самим як і поле "Пароль". Телефон і адрес мають такий самий сенс як і при заповненні інформації при замовленні у кошику рис. 5.4.

Після заповнення інформації, при успішній реєстрації, буде зображена сторінка з надписом "Вы успешно зарегистрировались, войдите", а при невдалому - буде видно повідомлення про те, що користувач з таким логіном чи паролем вже існує або невірно введене підтвердження паролю і можна буде повторити спробу.

5.2 Користувач

Користувач(User) вміє все те саме що і гість, але на етапі замовлення рис. 5.3 він може натиснути "Заказать" і йому не потрібно буде заповнювати інформацію, як користувачу, тобто замовлення буде на найближчий час і за його даними. Для того, щоб стати користувачем необхідно ввійти у систему, щоб увійти у систему потрібно перейти у розділ "Моя страница" і перейти по гіперпосиланню "войдите".

На рисунку 5.8 зображено як перейти до входу у систему

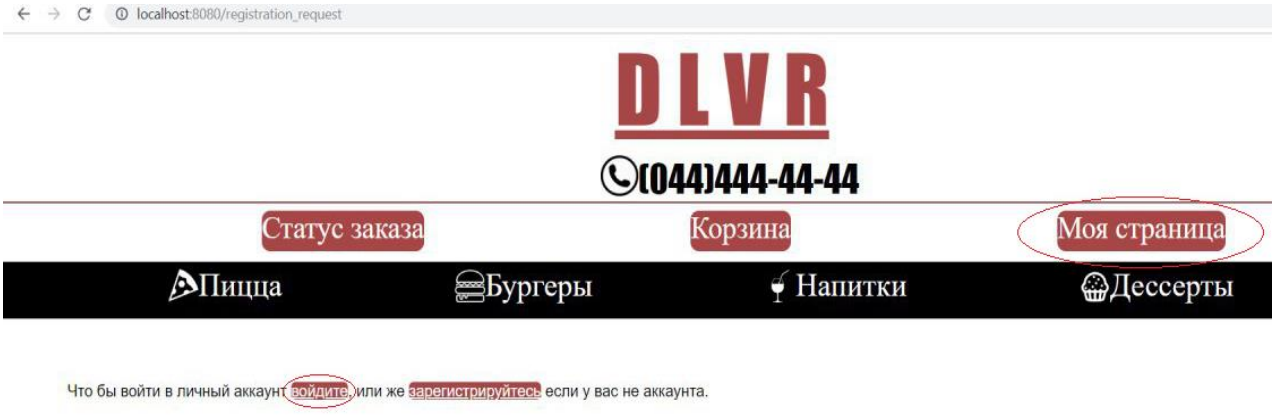


Рис. 5.8 Шлях до входу у систему

Після цього користувача буде напрямлено на сторінку з бланком, у якому є два поля: "Логин" і "Пароль" і кнопка "Войти". Після того, як користувач ввів свої дані, він перейде на сторінку з його власним акаунтом.

Рис 5.9 Власний акаунт

На цій сторінці можна редагувати свою власну інформацію, переглянути історію замовлень і зробити побажання.

Щоб продивитись історію замовлень, треба натиснути "История заказов" і браузер видасть наступне:

Рис 5.10 Історія замовлень користувача

Щоб залишити коментар або відгук, треба натиснути "Пожелания" і залишити коментар у полі і натиснути "Отправить":

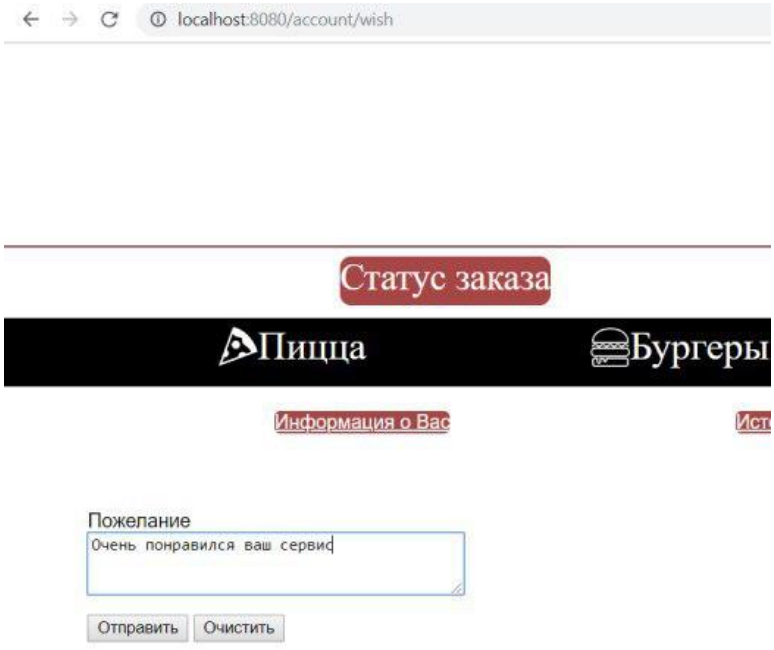


Рис. 5.11 Сторінка з побажанням

Також можна вийти із системи - кнопка "Выйти". Клієнта буде напрямлено на початкову сторінку.

5.3 Кухар

Задача кухаря полягає в тому, щоб брати замовлення, готувати його і відмічати замовлення приготовленим. Для того, щоб увійти у систему, йому потрібно перейти до бланку входу, як показано на рисунку 5.8, ввести логін і пароль, які йому видав адміністратор.

На рисунку 5.12 зображено власний акаунт кухаря

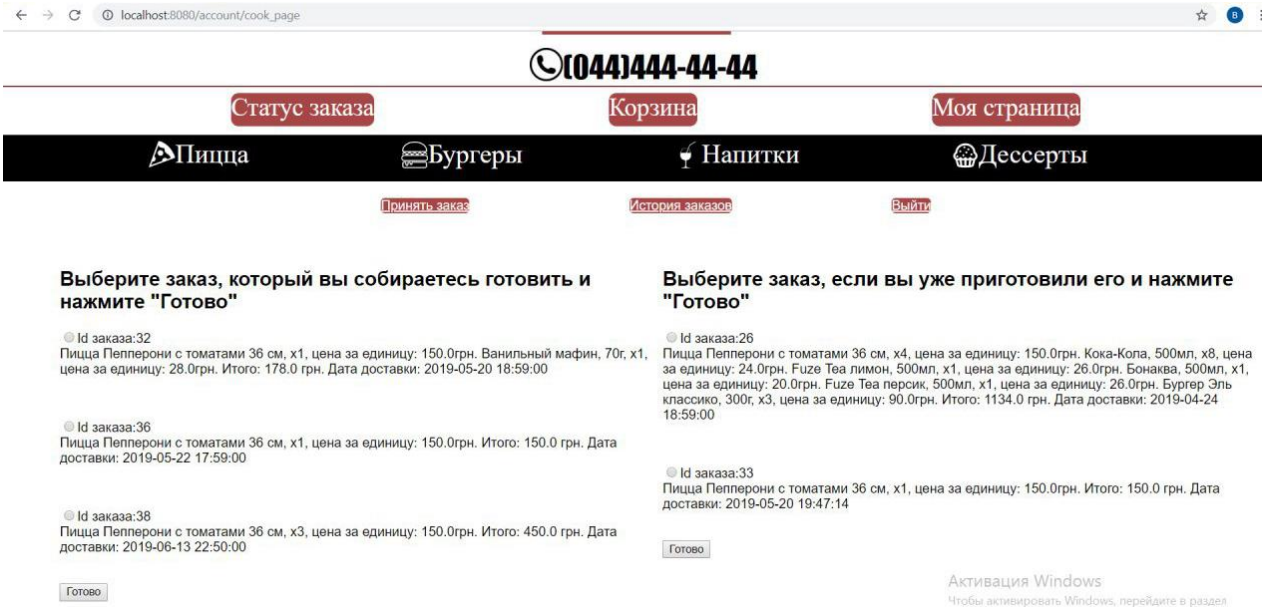


Рис 5.12 Власний акаунт кухаря

Зліва знаходяться замовлення зі статусом "ordered" і їх опис, тобто можна готувати, але тільки треба обрати одне з замовлень і натиснути "Готово".

Для прикладу візьмемо 38-е замовлення і відмітимо його як замовленням, що готується(cooking).

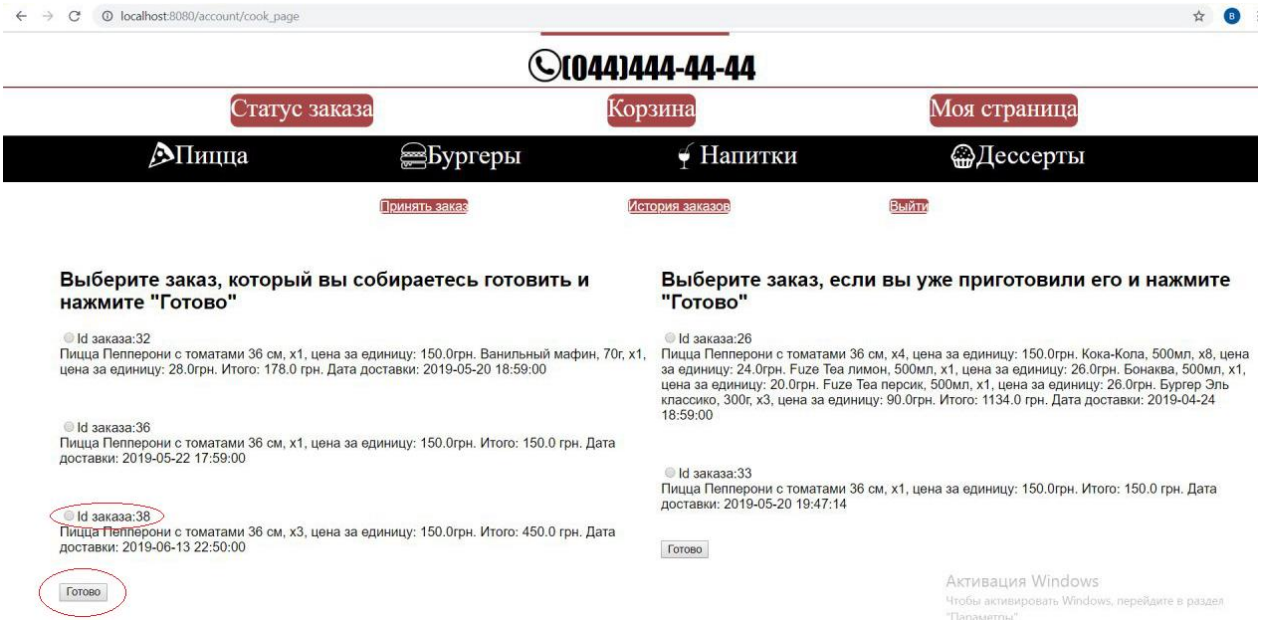


Рис. 5.13 Вибір замовлення

З рисунку 5.14 видно, що замовлення номер 38 перейшло у праву частину замовлень.

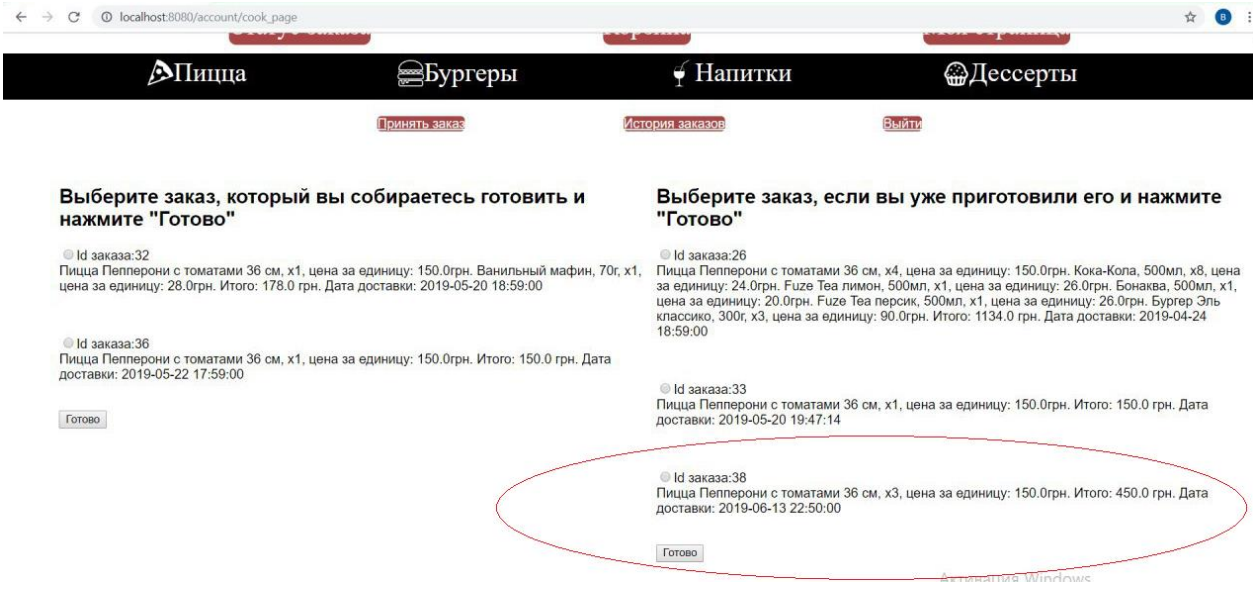


Рис 5.14 Демонстрація зміни статусу

Справа у кухаря замовлення, які він вже приготував, йому треба відмітити їх, як приготовленими(cooked), щоб адміністратор і кур'єр знали, що це замовлення можна вже доставляти.

Так само як на рисунку 5.13 кухарю потрібно відмітити замовлення приготованим, тобто обрати замовлення і натиснути "Готово".

Також кухар може переглянути історію всіх замовлень, які він готував, щоб їх подивитись, треба перейти по гіперпосиланню "История заказов".

На рисунку 5.15 зображено всі замовлення і їх статус, які готував кухар, який знаходиться у системі.

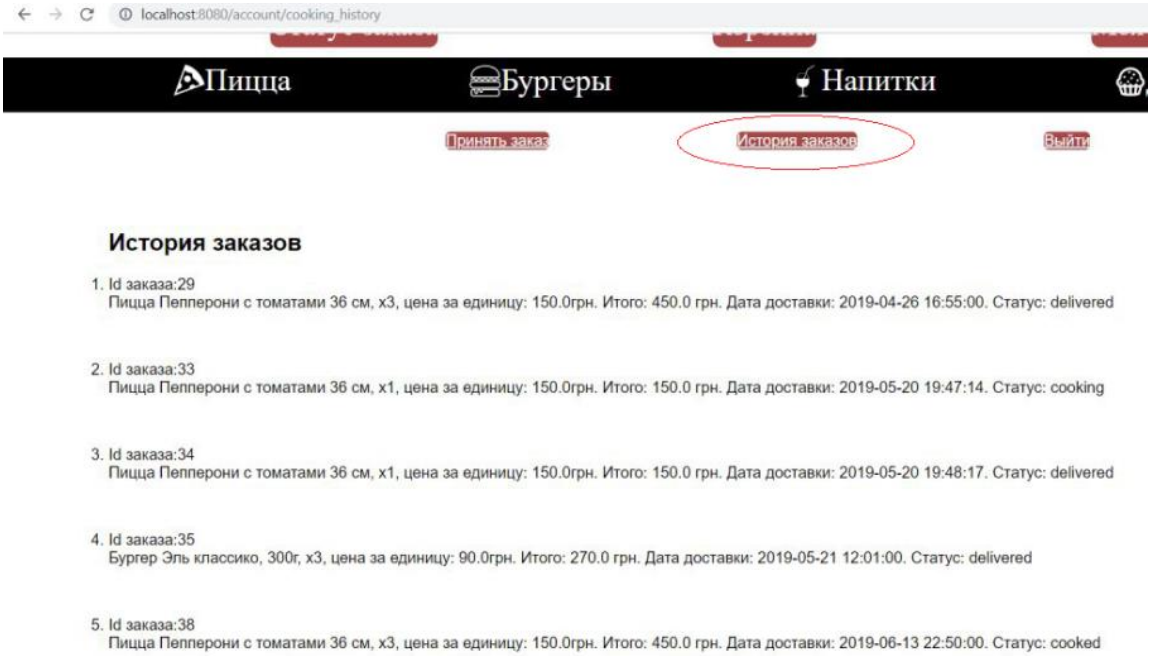


Рис 5.15

Кухар може вийти із системи, натиснувши "Выйти", він буде напрямлений на початкову сторінку веб-сайту.

5.4 Кур'єр

Задача кур'єра полягає у тому, що він має взяти замовлення, яке має статус cooked і доставити, по адресі, яку йому дасть адміністратор.

Щоб увійти у систему, йому потрібно так само як користувачу і кухарю перейти на сторінку з авторизацією, рис 5.8.

Коли кухар відмітив, що замовлення номер 38 приготовлене, кур'єр буде мати це замовлення у себе у списку зліва. Кур'єр так само як на рисунку

5.13 відмічає замовлення, яке буде доставляти. На рисунку 5.16 зображено це замовлення, яке було обрано для прикладу.

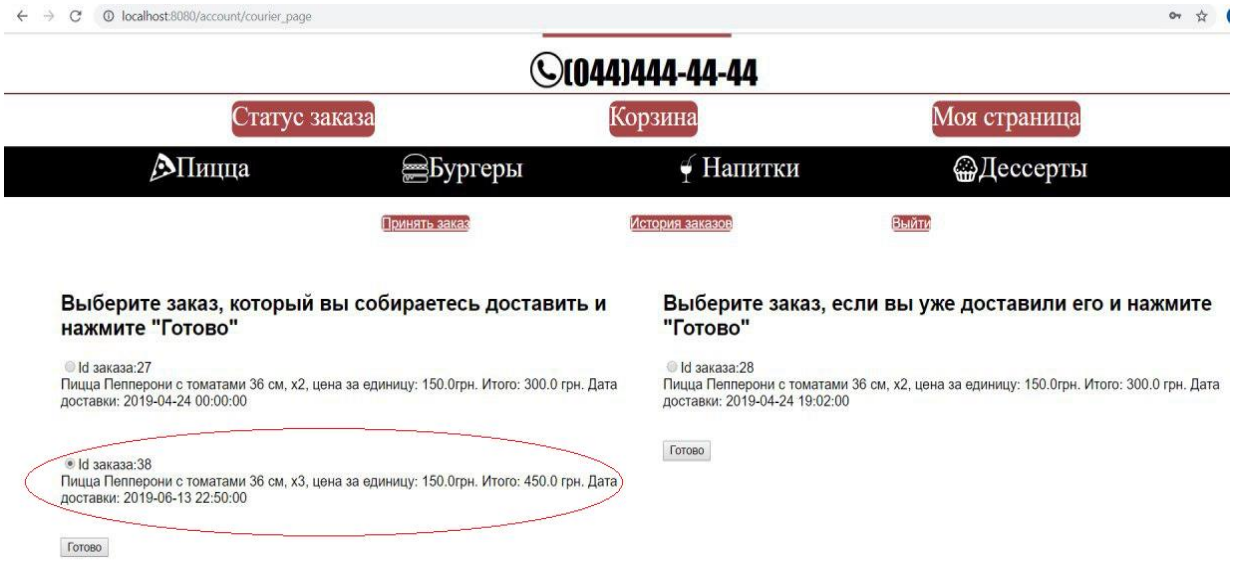


Рис 5.16

Коли кур'єр відмічає замовлення, яке буде доставлено, воно переходить у праву частину замовлень і має статус delivering. Так само як на рисунку 5.14. Відмічаючи замовлення номер 38 справа, кур'єр надає йому статус delivered, тобто замовлення є доставленим і оплаченим.

Так само як і на рисунку 5.15 кур'єр може подивитись замовлення, які він коли-небудь доставляв, а також вийти із системи, натиснувши "Выйти".

5.5 Адміністратор

Адміністратор може створювати кухаря і кур'єра, переглядати побажання користувачів, а також назначати на замовлення робітників.

Так само, як користувач, кухар і кур'єр, адміністратор має пройти авторизацію, рисунок 5.8. По завершенню авторизації, він перейде у свій акаунт, де він може виконувати свій функціонал.

Щоб створити кухаря або кур'єра треба перейти по гіперпосиланню "Создать повара" або "Создать курьера" відповідно.

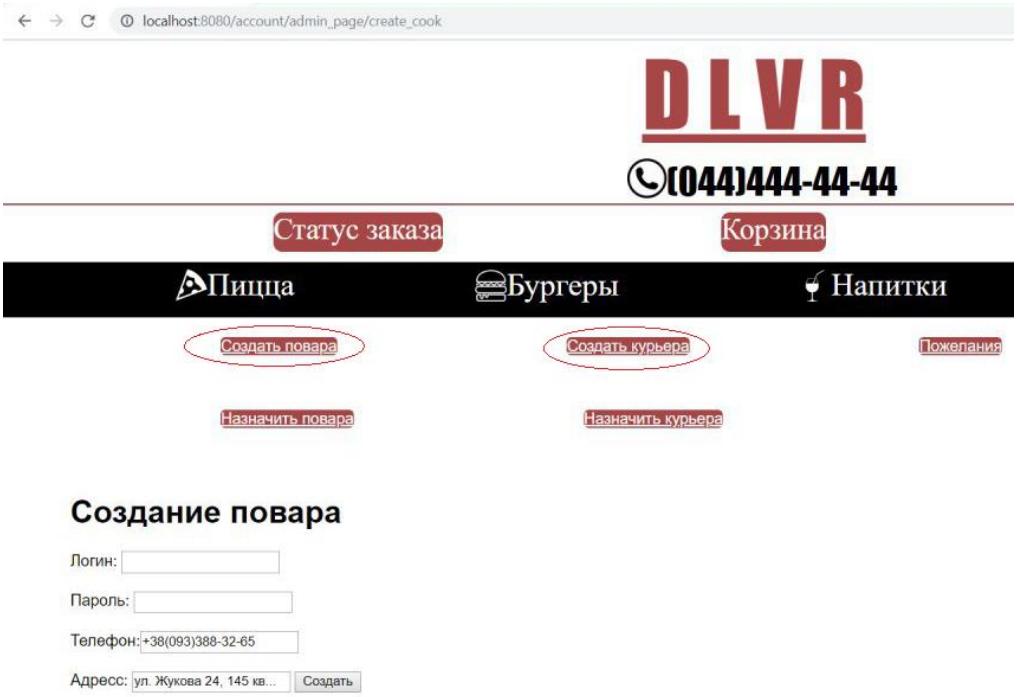


Рис 5.17 Бланк реєстрації кухаря або кур'єра

Адміністратор повинен наглядати за побажаннями, щоб задовольнити потреби користувачів і покращити сервіс, щоб перейти до сторінки з побажаннями, треба натиснути "Пожелания". На цій сторінці будуть побажання користувачів і їх ідентифікаційний номер.

На рисунку 5.18 зображені побажання, які були зроблені користувачами.

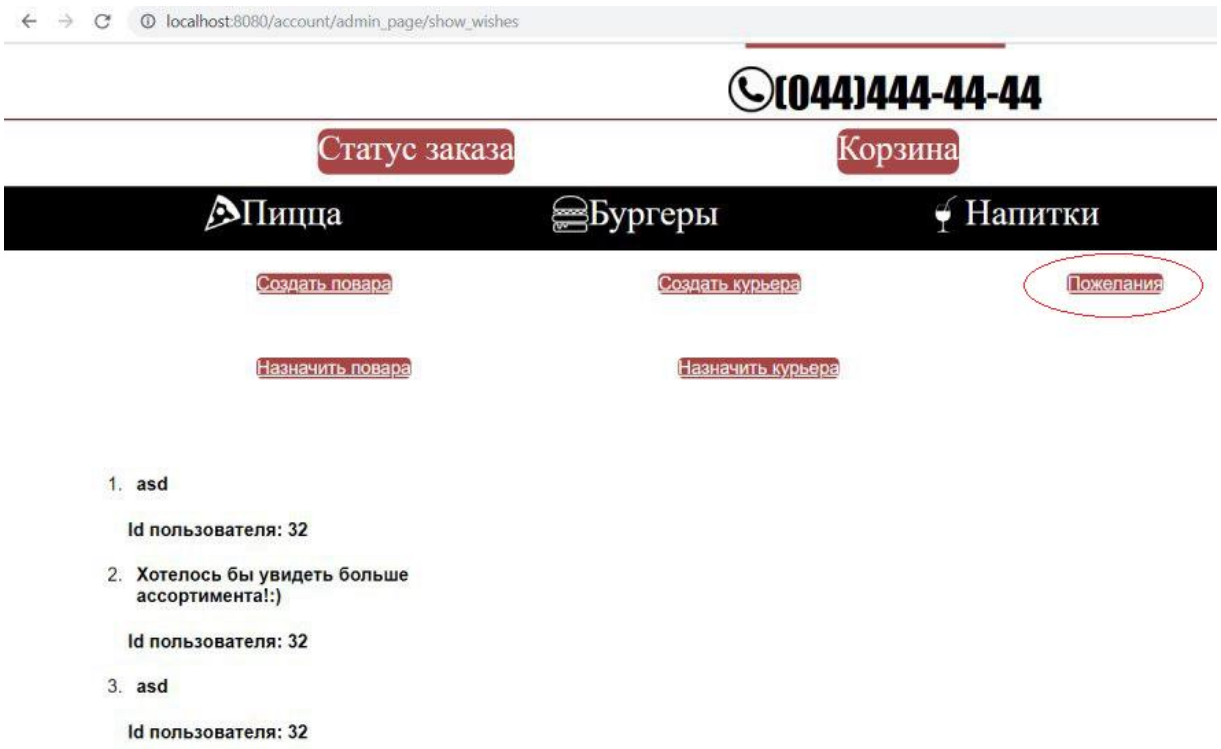


Рис. 5.18

Щоб назначити кухаря, адміністратор переходить у розділ "Назначить повара" і обирає замовлення й кухаря, який буде готувати замовлення і натиснути кнопку "Назначить", тобто адміністратор має список вільних кухарів і замовлень, які мають статус ordered.

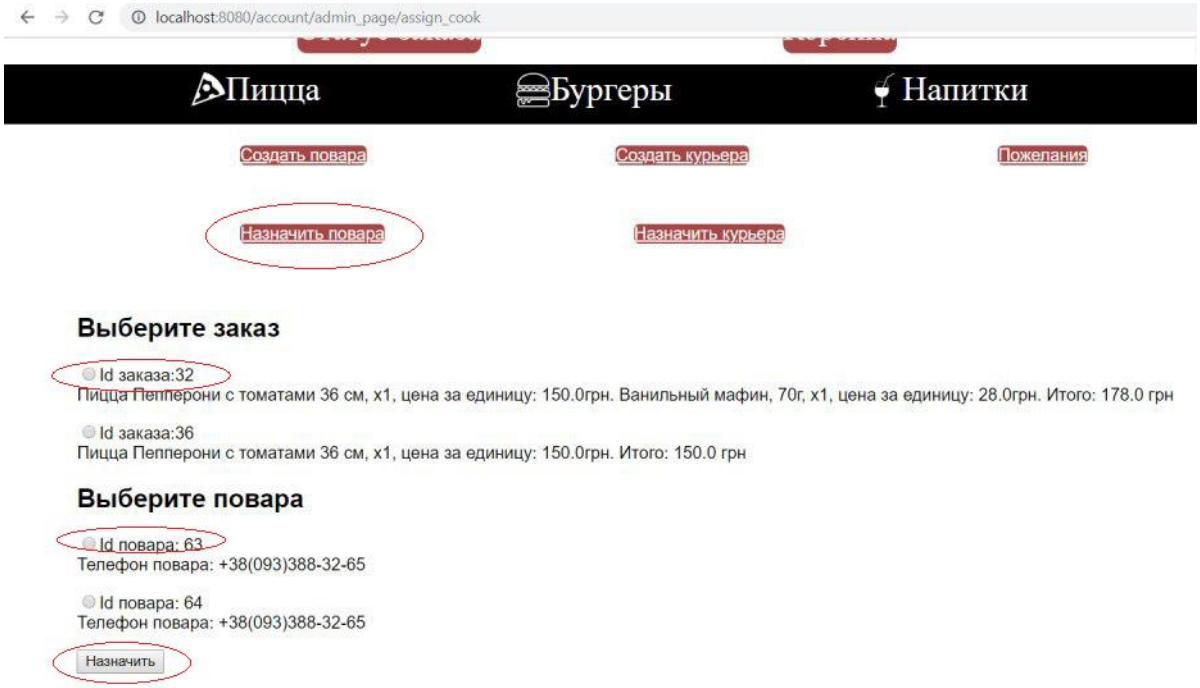


Рис 5.19 Сторінка призначання кухаря на замовлення

Так само і для кур'єра: на сторінці призначенні кур'єра адміністратор має вільних кур'єрів і замовлення, які мають статус cooked.

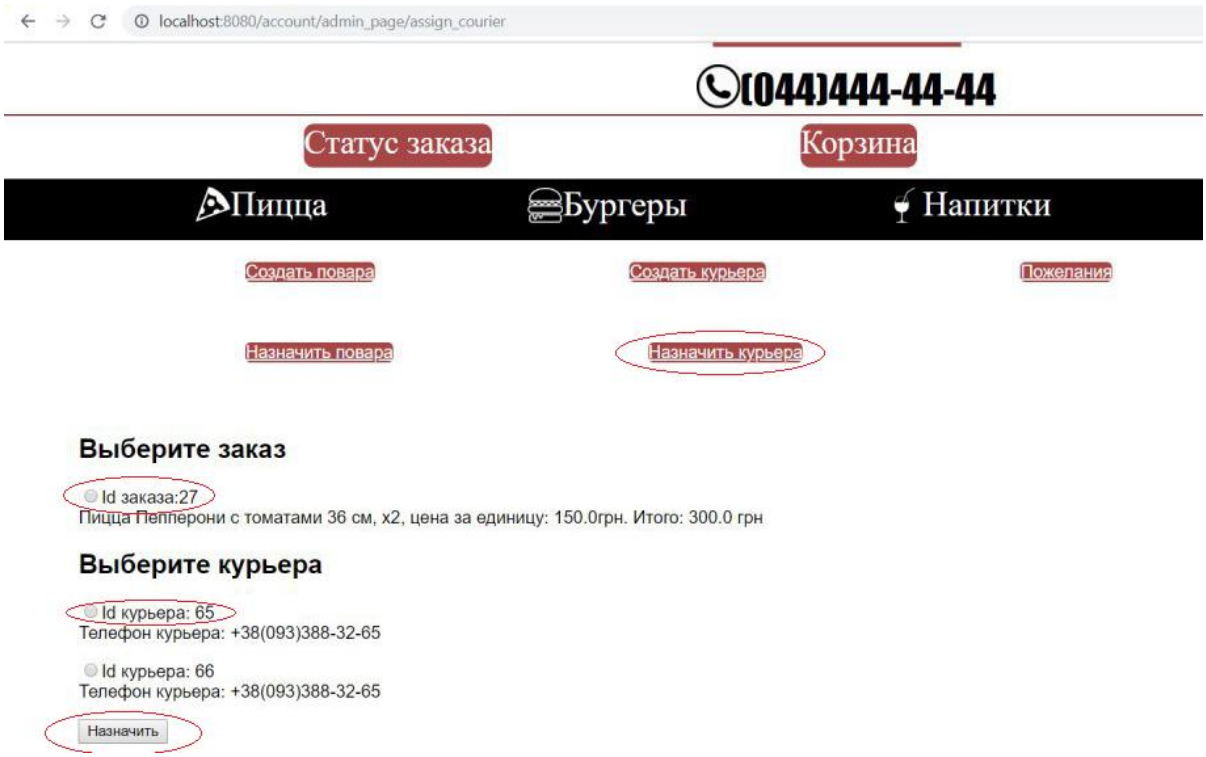


Рис 5.20 Сторінка призначання кур'єра на замовлення

По завершенню роботи адміністратор має змогу вийти із системи натиснувши кнопку "Выйти".

5.6 Висновок

Було проілюстровано роботу кожного типу клієнта на локальній адресі "localhost:8080". За допомогою рисунків був описаний весь функціонал, який є на веб-сайті, а також для більшого розуміння системи, були описані статуси замовлення.

ВИСНОВКИ

Під час виконання дипломної роботи були розглянуті сучасні рішення сервісів доставки їжі, була розроблена модель локального веб-сайту для надання сервісу клієнтам. Проаналізовано програмні мови та бібліотеки, які допомогли написати веб-застосунок, який задовольняє потребам дипломного проекту. Було побудовано модель сайту, щоб мати уявлення про те, як функціонує кожна роль у ньому.

Для реалізації дипломного проекту було виконано такі дії:

- Проаналізовано модернізовані сервіси доставки, зроблені висновки щодо кожного із них у вигляді недоліків і переваг;
- Вибір мови програмування і стиль архітектури для даного веб-проекту;
- Аналіз сучасних технологій створення сайтів і розбір бібліотек, для швидкого і якісного написання коду;
- Спроектовано концептуальну, функціональну модель і модель специфікацій для системи проекту;
- Описано роботу системи і їх ролі за допомогою рисунків зроблених з браузера.

Реалізований весь технічний функціонал, який необхідно використовувати для надання клієнтам сервісу доставки їжі.

Даний проект можна розширювати у майбутньому, а саме додати інтернаціоналізацію, локалізацію, інтегрувати програмний інтерфейс, який слідкує за геолокацією користувача, створити мобільний додаток під Android та IOS і написати API за допомогою REST архітектури.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура програмного забезпечення [Електронний ресурс] - Режим доступу до ресурсу: https://kpi.stu.cn.ua/arhitektura-programnogo-zabezpechenn/.
2. Клієнт-серверна архітектура [Електронний ресурс] - Режим доступу до ресурсу: https://buklib.net/books/23148/.
3. Мова програмування Java [Електронний ресурс] - Режим доступу до ресурсу: https://fainaidea.com/jeto-interesno-znat/pochemu-java-a-ne-chto-to-drugoe-130156.html.
4. Інтегроване середовище розробки (IDE) Eclipse [Електронний ресурс] - Режим доступу до ресурсу: https://hightech.in.ua/content/art-eclipse-platform.
5. Веб-розробка [Електронний ресурс] - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Веб-розробка.
6. Технології створення веб-сайтів [Електронний ресурс] - Режим доступу до ресурсу: https://knowledge.allbest.ru/programming/2c0a65625b3bc79a5c53a88421306d26_0.html.
7. Система керування контентом (CMS) [Електронний ресурс] - Режим доступу до ресурсу: https://hostiq.ua/wiki/cms/
8. Бібліотека Spring для Java [Електронний ресурс] - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Spring_Framework.
9. Spring Inversion of Control (IoC) [Електронний ресурс] - Режим доступу до ресурсу: https://proselyte.net/tutorials/spring-tutorial-full-version/spring-mvc-framework/.
10. Java Server Pages (JSP) [Електронний ресурс] - Режим доступу до ресурсу: http://java-course.ru/student/book1/jsp/.
11. Діаграма прецедентів [Електронний ресурс] - Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-use-case-diagram.

12. Діаграма класів [Електронний ресурс] - Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-class-diagram.

13. Діаграма діяльності [Електронний ресурс] - Режим доступу до ресурсу: https://flexberry.github.io/ru/fd_activity-diagram.html